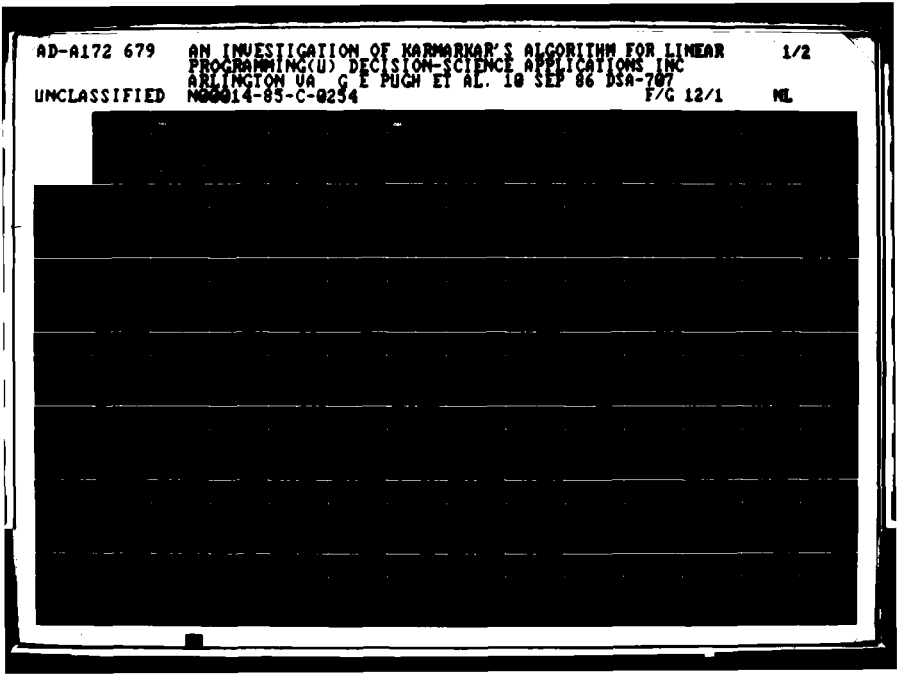
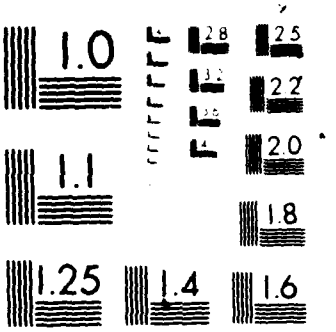
AD-A172 679 AN INVESTIGATION OF KARMARKAR'S ALGORITHM FOR LINEAR PROGRAMMING(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA G E PUGH ET AL. 10 SEP 86 DSA-707 1/2

UNCLASSIFIED N00014-85-C-0254 F/G 12/1 NL





(12)

DSA DECISION-SCIENCE APPLICATIONS

DSA Report No. 707

10 September 1986

AD-A172 679

AN INVESTIGATION OF KARMARKAR'S ALGORITHM FOR LINEAR PROGRAMMING

SUMMARY - FINAL REPORT

DTIC FILE COPY

prepared under contract #N00014-85-C-0254 for:

Office Of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Attention: Dr. Neal Glassman
Mathematics Department

86 9 15 008

Suite 1000, 1901 North Moore Street, Arlington, Virginia 22209 (703) 243-2500

393157 pgs

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A172 679

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: N/A
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): DSA Report #707	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Decision-Science Applications, Inc. / 6b. OFFICE SYMBOL (If applicable): 8N026	7a. NAME OF MONITORING ORGANIZATION: Office of Naval Research
6c. ADDRESS (City, State and ZIP Code): 1901 N. Moore Street, Suite 1000 Arlington, VA 22209	7b. ADDRESS (City, State and ZIP Code): 800 N. Quincy Street Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Office of Naval Research / 8b. OFFICE SYMBOL (If applicable):	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-85-C-0254
8c. ADDRESS (City, State and ZIP Code): 800 N. Quincy Street Arlington, VA 22209	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): An Investigation of Karmarkar's Algorithm for Linear Programming	
12. PERSONAL AUTHOR(S): G.E. Pugh, J. Danskin	
13a. TYPE OF REPORT: FINAL / 13b. TIME COVERED: FROM 4-1-85 TO 3-31-86	14. DATE OF REPORT (Yr., Mo., Day): 86 September 10 / 15. PAGE COUNT: 130
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): optimization; linear programming, non-linear programming, quadratic programming; Karmarkar, projective transformation, numerical stability, (cont. on reverse)

19. ABSTRACT (Continue on reverse if necessary and identify by block number):

During this contract period DSA has concentrated on achieving a geometrical rather than an algebraic understanding of Karmarkar's algorithm, and on the formulation of alternative approaches that retain the main advantages of Karmarkar's approach while avoiding the undesireable computational aspects of his specific algorithm.

From a geometric perspective, we have asked why does the algorithm work, when it works? How can one understand the convergence process, which occurs in the shifting coordinates of Karmarkar's simplex, in terms of an orderly convergence process in the original space? This has led to a number of non-obvious insights concerning the essence of the Karmarkar algorithm. The findings of this part of the study are contained in a separate report that has been distributed, and is being prepared for publication.

(continue on reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code): 703/243-2500 / 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Block #18 - balanced reduction of deficits, avoidance of matrix inversion.

Block #19 - In terms of algorithm improvements, we have searched for alternative approaches with better numerical stability so as to permit the use of approximate projection methods in place of the computationally burdensome matrix inversion. This search has led to a totally new approach to the LP problem which does not demand feasibility but rather accepts the infeasibilities as "deficits", using slack variables as required to make them equal--then seeking to drive them down in a "balanced reduction of deficits" in which they remain roughly equal. We have used various variants of Lemke's "closest-point principle" which, when properly applied, yields an approximate direction in Karmarkar's simplex corresponding to a proportional reduction of the deficits. Very recently, after the completion of the contract period, a method has been devised that does this in guaranteed time and thus promises a robust algorithm which avoids matrix inversion.

We have also found new theoretical results in quadratic programming which should make it possible to adapt the foregoing resutls to large-scale problems in quadratic programming. This work has led to an important paper that is now being submitted for publication.

DSA DECISION-SCIENCE APPLICATIONS

DSA Report No. 707 10 September 1986

AN INVESTIGATION OF KARMARKAR'S ALGORITHM FOR LINEAR PROGRAMMING

SUMMARY - FINAL REPORT

George E. Pugh

prepared under contract #N00014-85-C-0254 for:

Office Of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

Attention: Dr. Neal Glassman
Mathematics Department

Suite 1000, 1901 North Moore Street, Arlington, Virginia 22209 (703) 243-2500

TABLE OF CONTENTS

AN INVESTIGATION OF KARMARKAR'S ALGORITHM FOR LINEAR PROGRAMMING

SUMMARY FINAL REPORT

1.0 INTRODUCTION

1.1 SCOPE

This report summarizes the research on potential extensions and applications of the Karmarkar linear programming concept that was conducted by Decision-Science Applications, Inc. (DSA) for the Office of Naval Research (ONR) under Contract N00014-85-C-0254. The DSA research effort was initiated, with corporate funds, immediately after the publication of the original Karmarkar results, and has been continued with support provided by the Office of Naval Research under the above contract.

1.2 BACKGROUND AND OBJECTIVES

The testing and assessment of the Karmarkar algorithm by the mathematical community has made it clear both that the algorithm has a great deal of promise, and that the Karmarkar methods will require substantial additional development if they are to displace the simplex method as a standard technique for solving large-scale linear programming problems. As a result, considerable attention is being given to the development of more efficient methods for accomplishing the large matrix inversion operation which accounts for the bulk of the computational effort within the Karmarkar algorithm.

The present research effort has been focused somewhat differently on the possibilities for synergy between the Karmarkar concepts and the iterative Lagrangian optimization methods for large scale non-linear applications, which have been a DSA specialty. One remarkable aspect of the Karmarkar algorithm is that it defines an iterative computational approach for *linear optimization* that is similar in many ways to the iterative methods that are used in large-scale *non-linear optimization* applications. Thus, it seemed likely that the potential synergy between these two sets of optimization concepts could lead to new developments that might be more broadly applicable in the general field of mathematical optimization.

For this reason, the first objective of the DSA research effort has been to identify and clarify the *essential computational principles* that underlie the Karmarkar algorithm--with the objective of making them available as general purpose mathematical tools that can be more widely applied to mathematical programming problems outside the specific context of the Karmarkar algorithm.

The second objective of the DSA research has been to search for possible *alternatives* to the Karmarkar linear programming algorithm that might preserve its main computational advantages while *avoiding* entirely the computationally inefficient matrix inversion process.

We believe that substantial progress has been made with regard to both of the above objectives.

1.3 SYNOPSIS OF PUBLISHABLE RESEARCH FINDINGS

Analysis of Computational Principles

The DSA research has shown that the Karmarkar algorithm is mathematically equivalent to a simple steepest descent algorithm--in which "distance" is defined in terms of a *logarithmic distance metric* defined over the original problem variables, and in which the actual objective function is replaced by a *surrogate objective function* which includes a penalty function that facilitates boundary avoidance.

These basic findings are reported in *A Functional Analysis of the Karmarkar Linear Programming Algorithm*, George E. Pugh, DSA Report #676, March 31, 1986 (ref. 3.). This document, which is being transmitted with this final report, is also being distributed to the attendees at the ONR Monterey Conference (February 20-21, 1986) on the Karmarkar algorithm. It is expected that the main findings of the report will be prepared for publication in a mathematical journal.

Non-Matrix Inversion Alternatives

The development of alternative versions of the Karmarkar algorithm that can operate without matrix inversion appears to require two important departures from the basic Karmarkar algorithm. First, the basic algorithm must be restructured into a form that is less sensitive to inaccuracies in the calculated direction of steepest descent, so that approximation methods can be used in place of an exact matrix inversion. Second, computationally efficient and reliable methods for estimating the direction of steepest descent without matrix inversion must be developed.

A promising approximation method which was developed for estimating the direction of steepest descent without matrix inversion is described in another draft paper by John Danskin, "A Geometric Algorithm for Approximate Steepest Ascent with Linear Constraints," which is included as **Enclosure A** to this report.

One very promising approach which was developed to reduce the requirement for precision in the estimated direction of steepest descent is described in a draft paper by John Danskin, "A Variant of Karmarkar's Linear Programming Algorithm not Requiring Projection onto the Null Space," which is included as **Enclosure B** to this report.

1.4 OTHER RESEARCH FINDINGS.

As is usually the case in any exploratory research effort, the publishable material represents only a small percentage of the total effort, and some of the most interesting findings have not yet reached a point of development that would justify formal publication. The remainder of this report provides a broad overview of the research effort, and outlines some of the new insights which will need to be pursued in a continuation of the research effort.

2.0 TECHNICAL PROBLEMS WITH THE KARMARKAR ALGORITHM

About two years ago, Narendra Karmarkar proposed a radical new method for solving linear programming problems. The method involved the use of a projective transformation to map the space in which the problem was originally defined onto a simplex, in such a way that the current position point is mapped into the barycenter.

This method caused a sensation in the linear and non-linear programming communities. It seemed to be extremely fast on some problems, but extremely slow on others. There were difficult technical questions as to how to achieve feasibility and how to efficiently carry out the unfamiliar matrix inversion that Karmarkar requires; and questions as to the adequacy of the proof of convergence. But it is unquestionable that Karmarkar's basic idea is one of great power.

The main difficulty with the algorithm has been with the calculation of a direction of steepest descent within the simplex, which is required for each move. To accomplish this, Karmarker projects the objective vector into the "null space" defined by the constraints and the simplex equation. He accomplishes this projection algebraically by inverting a matrix.

There are many difficulties in inverting large matrices. J.A. Tomlin in "An Experimental Approach to Karmarkar's Projective Method for Linear Programming," which is included as **Enclosure C**, writes of problems encountered with Karmarkar's matrices. Their "condition" worsens as the iteration proceeds; also, any known inversion technique will decrease the sparsity of the matrix.

There are other problems. Computational inverses are always imperfect, and the Karmarkar algorithm is sensitive to computational inaccuracies. Without an exact inverse, the position vector will wander away from feasibility. Another difficulty is that of achieving an initial feasible point in the form Karmarkar wants it, with all the coordinates positive.

Finally, the Karmarkar proof of convergence hinges fundamentally on knowing the value of the function at the optimum. Karmarkar dealt with this problem in his original Bell Laboratory paper (ref. 1.) by using a rather awkward procedure to estimate the optimum value as the algorithm progressed. But he offered no proof of convergence for this procedure. In his revised paper in Combinatorica (ref. 2.), he solves the problem by using a combined primal-dual problem definition in which the value of the combined objective function at the solution point can be guaranteed to be zero. This approach provides a nice solution for the proof of convergence at the expense of using a much larger matrix in the computational process.

3.0 OVERVIEW OF DSA RESEARCH EFFORT

3.1 SPECIFIC RESEARCH OBJECTIVES

The DSA effort focused on two major objectives.

The first objective was to develop an improved understanding of the practical computational concepts that account for the performance of the Karmarkar algorithm--so that the concepts could be effectively utilized in the field of mathematical optimization outside the context of Karmarkar's specific algorithm. This objective has been essentially achieved, and the results are reported in DSA Report #676 (ref. 3).

The second objective was concerned with the development and study of possible variants of the Karmarkar algorithm, with the objective of improving the speed and reliability of the algorithm. This part of the DSA effort had several specific objectives. These included the development of:

- Approximate computational methods for finding steepest ascent that could be more efficient than matrix inversion, particularly because many such methods preserve matrix sparseness.
- Alternative algorithms for utilizing the Karmarkar steepest descent concept that are more tolerant of computational inaccuracy and departures from the precise null space than the original Karmarkar formulation.
- Alternative transformations of the original problem for accomplishing the same functional objective as the Karmarkar projection, but that are mathematically simpler and possibly computationally more efficient.

3.2 THE RESEARCH TEAM

To facilitate interactions between the new Karmarkar concepts and earlier iterative techniques developed for non-linear programming problems, DSA brought together a combination of theoretical and applied mathematical experience.

John M. Danskin, who is known for his extensive contributions to mathematical optimization and game theory, has served as the theoretical mathematician for the project. He was responsible for the original theoretical insights concerning the Karmarkar algorithm that led DSA to initiate the project, and he has contributed a substantial fraction of the theoretical concepts that have been investigated. In addition to his theoretical contributions, Danskin has also developed numerous Fortran programs on an IBM PC that he has used to test some of his theoretical concepts.

Most of the applied research for the project, including computer validation and large-scale testing of algorithms, has been conducted by Fred Miercort, who has more than 20 years of experience in applied mathematics and optimization methods. Miercort was responsible for most of the initial testing of variants of the Karmarkar concept that provided the foundation for a more theoretical functional understanding of the algorithm.

The project was supervised by George F. Pugh, who has many years of practical experience in solving very large non-linear optimization problems. In addition

to his role as supervisor for the project, Pugh was also responsible for developing the functional understanding of the algorithm as reflected in DSA Report #676 (ref. 3).

4.0 SUMMARY OF RESEARCH BY JOHN M. DANSKIN

4.1 EARLY RESEARCH CONTRIBUTIONS

Because of the promise of the Karmarkar approach and the numerous problems with the current Karmarkar formulation, Danskin began in November 1984 to seek methods for avoiding the exact-inverse, destruction of sparseness, and feasibility problems. He interested DSA in a joint research program on the topic, and continued with his investigations. Danskin made a number of very important early contributions to the research effort.

- He provided a detailed assessment and interpretation of the Karmarkar concept that provided the foundation for much of the DSA effort.
- He suggested the use of the Lemke closest point principle as a way of converting the calculation of a direction of steepest ascent into a quadratic distance minimization problem.
- He developed a number of approximate methods for solving the quadratic minimization problem, including a method of "maximum scoop" that has performed as well as, or better than, any other method that was tested by Miercort.
- He has explored several ways of utilizing primal-dual relationships to restructure standard LP problems so as to avoid some of the theoretical and computational problems of the Karmarkar approach.

4.2 PRESENT DANSKIN APPROACH

By June 1985, Danskin, after numerous excursions, arrived at a "balanced reduction of deficits" approach which seems particularly promising. In this approach, which is explained in sections 4 and 5 of Enclosure B, he uses the Karmarkar simplex and projective transformations, but otherwise departs radically from Karmarkar.

The approach never requires a feasible point. It starts, instead, with an arbitrary positive primal, dual pair (X,Y) which satisfies the "connecting equation" $C \cdot X = B \cdot Y$, but which is otherwise infeasible relative to many of the primal and dual inequality requirements.

The magnitude of the infeasibilities, measured positively, are referred to as "deficits." It may happen that the "deficits" corresponding to some of the inequalities are negative. Additive "slack" variables, somewhat analogous to the slack variables of the simplex method, are then introduced so as to make all the deficits positive and equal.

Starting with the deficits all equal, a new dummy variable, W, is introduced which is equal to them. The algorithm then seeks to find a way down for all the deficits which keeps them all approximately equal. On succeeding moves, any small deviations from the equality balance are partially corrected in such a way that, as W decreases towards zero, the deficits continue to remain within 25% of W. At no point is extreme accuracy required.

The method never requires a feasible point, and it obviates the need for an accurate projection. In a certain technical sense, the method is "projecting," but not in Karmarkar's sense.

The key to the success of such an approach, of course, lies in the availability of a computationally efficient method for estimating a direction that will maintain an approximate balance of the deficits during the convergence process. At the time the paper in Enclosure B was written, Danskin was using a new geometric (rather than algebraic) algorithm for steepest ascent, subject to equality constraints, which is included here as Enclosure A.

This algorithm works by always having a rate of ascent greater than the steepest constrained rate, and by gradually reducing the violations of the constraints. When this steepest ascent method is applied to the balanced reduction algorithm, directions are accepted as good even if the balanced reduction of constraints is violated by some moderate amount such as 25%. The resulting error is corrected in the next move.

More recently, Danskin has developed an approach which seems to perform much more efficiently, which is based on the application of a game solution concept which Danskin identifies as a "stack game." Regardless of what method is used to estimate a direction of balanced reduction, the method has the major advantage that it requires no operations at all on the matrix other than row multiplications and, thus, it completely preserves the sparsity of the matrix.

The two papers discussed above have been written under the current ONR contract. The corresponding computer programs, with some of the more recent techniques for estimating the move direction, have been used to solve linear programming problems with several thousand non-zero entries on an IBM personal computer--one that is not even equipped with hardware floating point capability.

4.3 A FAST POLYNOMIAL METHOD FOR REDUCING THE DEFICITS

Subsequent to the completion of the work on this contract Danskin has found a new method for finding a direction which drives the deficits down at exactly the same rate. The method takes at most $q - 1$ steps, where q is the number of constraints. Each step requires $pq\epsilon$ multiplications, where ϵ is the density of the non-zero elements. The method is an outgrowth of the approach given in section 2 of Enclosure B, but differs in the methods of solution defined thereafter.

It has now been incorporated in a running LP routine which has been developed on DSA's and Danskin's own time. An early version is included as Enclosure D. At the time that this draft was written, Danskin had not yet discovered that it is not necessary, in forming a direction for a move, to subtact off the projections on all the previous directions. One need only subtract off the projection on the last previous move. This discovery eliminated a factor of $q/2$, the required number of multiplications, and is the basis for the possibility that the new method will be really competitive in practice.

4.4 OTHER DANSKIN RESEARCH

Danskin has recently begun work on the application of these concepts to quadratic programming. He hopes to apply the balanced-reduction-of-deficits method to provide an iterative method for the solution of large quadratic programming problems. He has found a theorem which appears, after checking with leading experts, to be new, on "pseudo-duality," which makes it possible to convert a quadratic programming problem with linear constraints into a form that can be addressed by methods very similar to those outlined above.

5.0 EXPLORATORY RESEARCH BY MIERCORT AND PUGH

This exploratory research has focused on four major topics.

5.1 SIMPLIFICATION OF KARMARKAR TRANSFORMATION

The transformation Karmarkar uses to map the original LP problem onto a simplex would be linear except for his use of a denominator that makes the transformation non-linear. The presence of the denominator seems to introduce many complications into the algorithm. For example, it appears to be an important factor in motivating Karmarkar to work with a complicated surrogate objective function.

From an analysis of the effects of the distance metric on the direction of the Karmarkar moves, Pugh became convinced that Karmarkar's denominator (which was necessary for his proof of convergence) is not essential to the operation of the algorithm. To test this conjecture, Miercort developed and tested a simplified algorithm which does not use a surrogate objective function, and which omits the denominator from all calculations.

Although no implementation of the full Karmarkar algorithm was available at that time with which we could make direct comparisons, the operation of the simplified algorithm seemed indistinguishable from what had been reported by other investigators using the full Karmarkar transformation. About the same number of moves were required to achieve convergence on comparable problems. The trade-off between the number of moves and the size of a move was the same as reported by other investigators, and the same types of problems were encountered with the matrix inversions required to solve the steepest ascent sub-problem. We learned from the ONR Monterey meeting that we had not been alone in this approach. One speaker there listed more than a dozen similar "affine transformation" approaches.

Subsequent research (ref. 3) has shown that the omission of the surrogate objective function does somewhat reduce the efficiency of the Karmarkar algorithm. But the differences in the performance were too small to be apparent at that time without a direct side-by-side comparison. The subsequent research has also suggested, however, that the use of the simplified distance metric without the denominator may actually be somewhat more efficient than the full Karmarkar distance metric.

5.2 A TEST BED TO INVESTIGATE COMPUTATIONAL ACCURACY PROBLEMS

DSA's experimental work with various forms of this simplified algorithm focused on number of test problems, ranging from only a few variables to several hundred. Tests were made using both sparse matrices, that are characteristic of most commercial linear programming, and dense matrices that are common in game theory optimization problems.

Since the basic Karmarkar approach involves an iterative computational algorithm, one might reasonably expect that it would be self-correcting and tolerant of computational inaccuracy. Indeed, Karmarkar himself, in his original paper, made just such a claim. But the experience of all other investigators has been that very high precision is required in the matrix inversion to avoid drifting away from the feasible

null space in a succession of the prescribed Karmarkar moves. Moreover, such drift is not self-correcting in the basic algorithm, and the difficulty becomes more severe as the algorithm approaches the solution point.

To facilitate an investigation of problems in computational accuracy, the "test-bed algorithm" was designed so that iterative approximate methods could be used interchangeably with the more standard matrix inversion methods. Moreover, the test-bed was so structured that the performance of each iterative method could be measured relative to the more accurate matrix inversion results.

Two basic types of test problems were used in the analysis. With the cooperation of KETRON Inc., a company that is very active in commercial linear programming, several typical test problems have been obtained, ranging from about 10 to about 300 variables. Like most commercial linear programming problems, they tend to be quite sparse. But they are also problems that include pitfalls that can be troublesome for simple linear programming codes. The second basic type of problem arises in the solution of game theory problems. We have concentrated particularly on the so-called "cookie-cutter" game, whose matrix is generally quite dense. The formulation of this game allows an easy selection of a wide variety of problems of varying size whose solution is very non-trivial.

5.3 EVALUATION OF ALTERNATIVE ITERATIVE ALGORITHMS

We have since assessed the performance of a wide variety of iterative algorithms. So far, none of the methods tested has performed well enough to be a serious competitor to the matrix inversion method, except perhaps on problems much larger than those we have used in the experimental work.

Almost all of the iterative methods that have been tested utilize versions of the Lemke principle, which converts the steepest ascent problem into a quadratic distance minimization problem. This method tries to find that linear combination of the constraint vectors which most accurately approximates the objective function. The vector coefficients that are produced in this way are analogous to the Lagrange multipliers that are encountered in non-linear programming problems. As the algorithm approaches the solution point, these variables converge to the values of the dual variables.

The methods explored include:

1. Well-known heuristic methods, due to Hugh Everett, that are used to adjust the Lagrange multipliers for non-linear programming problems;[1]

2. A simple steepest descent algorithm;

3. Various steepest descent algorithms, altered to include heuristic learning mechanisms;

---

[1] These heuristic methods simply raise a multiplier by some factor, $1+\alpha$, if the constraint is violated in an unconstrained Lagrange optimization, and lower it by a comparable factor, $1-\beta$, if the constraint is slack. The factors, α and β, by which a given multiplier is adjusted are arbitrarily increased when successive adjustments are in the same direction and are is decreased when a reversal of direction occurs.

4. A steepest quadratic descent algorithm developed by John Danskin, which uses a curved (i.e., quadratic) trajectory to correct for the first order changes in the direction of steepest descent; and

5. A "maximum scoop" minimization algorithm (also developed by Danskin), which uses a steepest descent method, in the space of directions, to search for a direction vector with the lowest trajectory minimum.

Most of these algorithms have also been tested on renormalized representations of the linear programming problem (in which the units of the constraints and the activities have been modified to provide a natural distance metric for defining a direction of steepest descent).

We have not had the resources under this contract to test some of Danskin's more recent direction-finding algorithms, such as the algorithm described in Enclosure A or his newest algorithm, which has generated exceptionally good performance on an IBM PC. This newest algorithm, which looks promising, utilizes a two-level scoop method in combination with the "stack game" optimization concept.

With the possible exception of the most recent "double scoop" concept, none of the above algorithms has yielded the uniformly reliable performance on all of the test problems that would be desirable in a standardized optimization program. However, the basic structure of the direction- finding problem is sufficiently predictable to justify the expectation that a very satisfactory iterative method can be identified that can be efficiently used in combination with an alternative formulation of the Karmarkar algorithm, and that is less sensitive to inaccuracies in the direction-finding process. Clearly, additional research on this topic is badly needed.

In this connection, we note that, as this report is being prepared, Danskin has found a polynomial algorithm for an "exact way down" without matrix inversion. We are, however, submitting this report before that algorithm, found on "own time," has been tested.

5.4 METHODS TO REDUCE VULNERABILITY TO COMPUTATIONAL INACCURACIES

This section discusses some of our approaches intended to reduce the vulnerability of the Karmarkar approach to computational inaccuracies.

Advance Solution of Equality Constraints. The difficulty of finding move directions that can preserve feasibility is greatly exacerbated by the presence of equality constraints. In commercial linear programming codes, it has become common practice to eliminate the equality constraints by solving the equations in advance. When any variables associated with equality constraints are eliminated, one may of course need to introduce additional inequality constraints to confine the solution space to regions where the eliminated variables have positive values.

From our preliminary tests, it appears that such a removal of the equality constraints may be even more important for a Karmarkar type of algorithm than it is for the standard simplex methodology. Thus it may be appropriate to begin by converting any LP problem into a canonical form, from which the original equality constraints have all been removed, before applying a Karmarkar-like optimization algorithm.

(Of course, the Karmarkar methodology requires the introduction of slack variables so that all inequalities are ultimately converted into equalities. However, the slack type of inequalities are computationally less troublesome, because each one is directly related to an inequality constraint--and the Karmarkar solution process takes place in a feasible region where these constraints are not rigidly binding.)

Infeasibility Penalty Functions. The use of penalty functions that can be added to the LP objective function, to encourage moves that tend to remove accumulated feasibility errors, is one possible approach for minimizing the vulnerability of the Karmarkar method to computational inaccuracies.

In this connection, we have examined two basic strategies. In one, the moves alternate between those driven solely by the feasibility consideration and those driven solely by the optimality objective. In the other, both considerations are combined in each of the moves. Both methods seem to work, but neither has shown a clear superiority.

A Primal-Dual Bootstrap Method. Experience with approximate solution methods for the Karmarkar algorithm has shown that, in many cases, the dual variables (that are a by-product of the solution of the steepest ascent sub-problem) seem to converge much more rapidly toward a solution than do the primal variables. This suggested an approach in which both the primal and dual representations of a problem are simultaneously solved, so that the dual variables that are calculated for each representation can be used (in what we have labeled a "bootstrap" move) to update the primal variables for the other representation. This approach has proved to be spectacularly successful in many cases. It not only moves rapidly toward optimality, but also corrects most of the accumulated feasibility errors. But in other cases, the method totally fails to converge.

We are trying to understand the reasons for the unpredictability of the method, and to develop indicators that can be used to determine when a "bootstrap" move is likely to be superior to a Karmarkar move.

Assessment of Alternatives. The alternatives noted above for maintaining exact or approximate feasibility within the Karmarkar algorithm will ultimately have to be evaluated in competition with techniques such as the Danskin reduction of deficits method--techniques which do not require feasibility in the computation phase, and which only approach feasibility in the final stages of the solution process.

Based on our work to date, we feel the advance solution of equality constraints is the most promising we have tried. The use of a feasibility penalty function remains an interesting option, while the boot-strap method appears to be fundamentally flawed.

6.0 FUTURE RESEARCH OBJECTIVES

This section summarizes research objectives for the immediate future.

- Convergence Proof for Simplified Distance Metric. The analysis in Ref. 3 showed that the Karmarkar method is equivalent to a simple steepest descent algorithm in which distance is measured in a logarithmic distance metric defined over the original variables. The analysis suggests that Karmarkar's convergence proof should be directly applicable to an alternative algorithm which uses the Karmarkar surrogate objective function in combination with the logarithmic distance metric--equivalent to what would be obtained by omitting the Karmarkar denominator. That this is so remains to be proved.

- Convergence Proof Without Surrogate Function. Computational experience has shown that remarkably good performance, comparable to the Karmarkar algorithm on most of our test problems, can be obtained with a much simpler algorithm which ignores the surrogate objective and uses a simpler distance metric--equivalent to omitting the denominator in the Karmarkar transformation. However, we have as yet no formal proof of convergence for this process.

- Performance Test of New Danskin Method. The new Danskin method is still being debugged. As it is being programmed and tested, important improvements in the method are being made. When these are complete, the method will need to be tested against alternatives already available.

- Analysis of "Bootstrap" Move. Theoretical analysis is needed to assess the circumstances under which the "bootstrap" move can be expected to be successful, and to develop indicators that can be used during computation to decide when such a move is appropriate.

- Improved Quadratic Minimization. Additional work is badly needed on the quadratic minimization sub-problem. We believe that this aspect of the problem is crucial, particularly for very large problems.

- Quadratic Minimization Methods for Other Problems. The iterative quadratic minimization methods that are being developed and tested for use within the Karmarkar algorithm have a wide variety of potential applications to other practical problems, such as regression analysis, the optimization of electric power networks, and other quadratic programming problems. Work is needed to assess the degree to which such methods might be useful for large-scale problems in these areas.

- Application to Non-Linear Programming. The Karmarkar concept has many features that are particularly attractive for non-linear programming applications. Many of these applications--in game theory, economics, and optimal control--now use either linear programming or Lagrange multiplier optimization methods to solve an optimization sub-problem that is later adjusted to match non-linear external conditions through an external iterative loop. In such applications, the linear programming

algorithm appears as a sub-routine that is called repeatedly during the course of the overall optimization, i.e. "inside a DO loop."

In such a system, each successive sub-problem will typically differ only slightly from the previous one. But the standard simplex methodology benefits surprisingly little from the availability of an approximately correct existing solution. For this reason, such systems are sometimes designed to use iterative Lagrange multiplier methods rather than the simplex methodology. But because of the iterative nature of the Karmarkar approach, it should benefit from the availability of an approximate solution in much the same way as the Lagrange multiplier methods. Thus, if it provides a reliable method of solution, it might reasonably be expected to displace the simplex method in many such applications.

REFERENCES

1. *A New Polynomial-Time Algorithm for Linear Programming*, N. Karmarkar, AT&T Bell Laboratories, Murray Hill, N.J. (as presented at the Symposium for the Theory of Computing, Washington, D.C., in April 1984).

2. "A New Polynomial-Time Algorithm for Linear Programming", N. Karmarkar, as published in *Combinatorica*, 1985.

3. *A Functional Analysis of the Karmarkar Linear Programming Algorithm*, George E. Pugh, DSA Report #676, March 31, 1986.

ENCLOSURE A
A GEOMETRIC ALGORITHM FOR APPROXIMATE STEEPEST ASCENT WITH LINEAR CONSTRAINTS

BY JOHN M. DANSKIN

ABSTRACT

The paper presents a new method for approximating to steepest ascent constrained by equalities. The time for convergence to accuracy ϵ is for a given problem proportional to $\log(1/\epsilon)$, but the proportionality constant in the estimate given can be very large. The method also applies to the problem of minimization of quadratic forms given as sums of squares of linear forms in all of Euclidean space without boundaries.

§1. Introduction

Consider the problem, stated for $a, b, \gamma \in R^p$:

Maximize $a \cdot \gamma$ (1.1)

subject to

$$|\gamma| = 1 , \quad b_j \cdot \gamma = 0 , \quad j = 1, \ldots, q . \tag{1.2}$$

Denote the maximum in (1.1)-(1.2) by v . We will replace this problem by an ϵ-problem, in which we seek, for a given positive ϵ , either to be told that $v \leq \epsilon$ or else to be provided with a γ satisfying

$$a \cdot \gamma \geq (1 - \epsilon)v \tag{1.3}$$

and

$$|\gamma| = 1 , \quad |b_j \cdot \gamma| < \epsilon , \quad j = 1, \ldots, q . \tag{1.4}$$

We present here an algorithm, geometric in nature, for producing a solution to the ϵ-problem. It has the disadvantage of not being

exact, but the advantage that it will work on any matrix, and without affecting its sparsity.

We have a convergence proof, but no speed estimate in terms of p, q, and ϵ other than that, for a given problem, the time required to achieve accuracy ϵ is proportional to $\log(1/\epsilon)$. We have tried crude predecessors of this method on small problems with encouraging results, and are certain that this will be much faster. We intend, before this paper goes to press, to have tested the method on large problems. Our particular application is to a much-modified version of the Karmarkar linear programming algorithm, in which we can do with rather large ϵ.

We explain the idea behind the algorithm in §2. It comes down to finding the closest point to the origin on a linear manifold. In §3 we describe the algorithm itself. In §4 we analyze the algorithm and prove that it converges. In §5 we note an application to the minimization of a class of quadratic forms.

§ 2. The idea of the algorithm

We may suppose that $|a| = 1$. If all of the b_j are orthogonal to a, then obviously $\gamma = a$ solves the problem. If some of the b_j are orthogonal to a and some are not, we alter those which are by adding a vector δa, where $\delta \in (0, \epsilon/2)$; this will not change any $b_j \cdot \gamma$ by as much as $\epsilon/2$. So from now on we assume that no b_j is orthogonal to a. Put

$$x^j = \beta_j b_j , \quad \underline{\text{where}} \quad \beta_j = 1/(a, b_j) . \tag{2.1}$$

Then $a \cdot x^j = 1$ for all j. Now denote by $\mathcal{L}$ the linear manifold of combinations

$$x = \sum \lambda_j x^j , \quad \underline{\text{where}} \quad \sum \lambda_j = 1 . \tag{2.2}$$

Then $a \cdot x = 1$ for all $x \in \mathcal{L}$. Evidently $\mathcal{L}$ does not pass through the origin.

Suppose we know the point c on $\mathcal{L}$ closest to the origin. Put

$$h = a - c/c^2 . \tag{2.3}$$

If $h = 0$, then, for any γ satisfying (1.2),

$$a \cdot \gamma = (1/c^2)\, c \cdot \gamma = (1/c^2) \sum \lambda_j^c x^j \cdot \gamma = \sum \lambda_j^c a_j (b_j \cdot \gamma) = 0 \qquad (2.4)$$

where the λ_j^c are the coordinates of c. Hence in that case $v = 0$. Otherwise put

$$v = |h| , \quad \gamma = (1/v)\, h . \qquad (2.5)$$

Since c is normal to any line in $\mathcal{L}$, then $c \cdot (x - c) = 0$ for any $x \in \mathcal{L}$. Hence $c \cdot x = c^2$ on $\mathcal{L}$; in particular $c \cdot x^j = c^2$ for any j. Using this, we see that

$$h \cdot x^j = a \cdot x^j - (1/c^2)\, c \cdot x^j = 1 - 1 = 0 \qquad (2.6)$$

for all j. It follows that γ satisfies the constraints (1.2). Next suppose that γ' satisfies (1.2). Then $c \cdot \gamma' = 0$, so that

$$a \cdot \gamma' = (a - c/c^2) \cdot \gamma' = v\gamma \cdot \gamma' . \qquad (2.7)$$

This implies that γ yields the unique maximum in the problem (1.1)-(1.2), and that that maximum is v.

The matter thus comes down to finding the closest point c, or a good approximation to it. That is what our algorithm does.

§ 3. The algorithm

We suppose the routine has arrived at a λ and corresponding x satisfying (2.2). It first tests to see whether we have sufficient evidence to assure that $v \leq \epsilon$. It does this by calculating

$$h = a - x/x^2 \tag{3.1}$$

with the x at hand rather than with c as at (2.3). If $|h| \leq \epsilon$, then $v \leq \epsilon$ and the routine terminates. If $|h| > \epsilon$, it forms

$$\gamma = h/|h| \tag{3.2}$$

and tests the inequalities in (1.4), with ϵ replaced by $\epsilon/2$ if the b_j has been altered as at the beginning of §2 to assure nonorthogonality with a. If all of those tests pass, it terminates. Otherwise it prepares for a move.

That it is not necessary to test (1.3), and that the above $|h| \leq \epsilon$ test is correct, is proved at Lemma D in §4.

The square of the distance from x to the origin is

$$Q(\lambda) = \sum_{i=1}^{p} \Big(\sum_{j=1}^{q} \lambda_j x_i^j \Big)^2 \quad . \tag{3.3}$$

The partials with respect to the λ_j are

$$Q_j(\lambda) = 2\mathbf{x} \cdot \mathbf{x}^j \quad , \quad j = 1, \ldots, q \quad . \tag{3.4}$$

The routine averages the $Q_j(\lambda)$, reverses the signs, and forms the direction numbers

$$2\mathbf{x} \cdot (\bar{\mathbf{x}} - \mathbf{x}^j) \quad , \quad j = 1, \ldots, q \, , \tag{3.5}$$

in which $\bar{\mathbf{x}}$ is the barycenter of the $\mathbf{x}^j$. It then calculates

$$\rho = 2 \Big[\sum_{j=1}^{q} (\mathbf{x} \cdot (\bar{\mathbf{x}} - \mathbf{x}^j))^2 \Big]^{1/2} \quad . \tag{3.6}$$

Because some of the tests have failed, $\rho > 0$. It now forms the direction cosines

$$\sigma_j = (1/\rho) 2\mathbf{x} \cdot (\bar{\mathbf{x}} - \mathbf{x}^j) \quad , \quad j = 1, \ldots, q \tag{3.7}$$

in the λ-space, moves to the bottom of the parabola in that direction, and proceeds.

§4. Analysis of the algorithm

The first task of this analysis is to examine the meaning of the quantity ρ in (3.7), which is the steepest rate of descent relative to the λ, as it relates to the geometry of the x-space.

We will show that a large ρ guarantees a good move, and a small ρ indicates that we are near the solution. These facts, proved as Lemmas B and C, immediately yield the convergence theorem. Lemma C may fairly be regarded as difficult.

The second task, very much easier, is to establish the facts about the tests involving the value which we used in the algorithm. This we do in Lemma D, after quoting a principle due to Lemke.

Put

$$w^j = \bar{x} - x^j , \quad j = 1, \ldots, q , \tag{4.1}$$

and then

$$\Delta = [\sum (w^j)^2]^{1/2} . \tag{4.2}$$

We suppose that $\Delta > 0$; otherwise the problem is trivial.

LEMMA A. Denote by ds and $d\sigma$ the lengths of the same infinitesimal segment on $\mathcal{L}$, measured relative to the x-space and λ-space respectively. Then

$$\frac{ds}{d\sigma} \leq \Delta . \tag{4.3}$$

PROOF. Fix on any λ^0 and corresponding x^0 . Consider an admissible direction issuing in $\mathcal{L}$ from λ^0 , i.e. a unit vector σ with

$$\sum \varrho_j = 0 \ . \tag{4.4}$$

Move a distance $\sigma > 0$ from λ^0 along ϱ. The corresponding distance relative to x is

$$s(\sigma) = \sigma \Big[\sum_{i=1}^{p} \Big(\sum_{j=1}^{q} \varrho_j x_i^j \Big)^2 \Big]^{1/2} \ . \tag{4.5}$$

Using (4.4), we may replace the term x_i^j in (4.5) by w_i^j. Call the result (4.5)'. From the Schwarz inequality,

$$\Big[\sum_{j=1}^{q} \varrho_j w_i^j \Big)^2 \le \sum_{j=1}^{q} (w_i^j)^2 \ , \ i = 1, \ldots, p \ . \tag{4.6}$$

On putting this into (4.5)' and reversing the order of summation, we get

$$s(\sigma) \le \Delta \sigma \ . \tag{4.7}$$

which proves (4.3). □

Now we will prove that a large ρ guarantees a good move.

LEMMA B. <u>If $\rho > 0$, a move in the λ-space direction ϱ given by (3.5) will yield a decrease of at least $\rho^2/4\Delta^2$ in the squared distance to the origin.</u>

PROOF. The directional derivative of Q in the direction g, relative to λ,

$$\frac{dQ}{d\sigma} = -\rho \quad . \tag{4.8}$$

We may rewrite this as

$$\frac{dQ}{ds} \times \frac{ds}{d\sigma} = -\rho \quad . \tag{4.9}$$

By Lemma A, i.e. (4.3), this implies that the directional derivative of Q in the x-direction corresponding to g satisfies

$$\frac{dQ}{ds} \leq -\rho/\Delta \quad . \tag{4.10}$$

Supposing that dQ/ds in that direction at x^0 is exactly $-\zeta$, we see that the formula for the squared distance along it is

$$Q_s = Q_0 - \zeta s + s^2 \quad . \tag{4.11}$$

s being the distance in $\mathcal{L}$ relative to x. Here the reader should recall that the second directional derivative of the squared-distance function Q, relative to x, in any direction, is 2. The minimum of Q is attained at $s = \zeta/2$, and its value is $Q_0 - \zeta^2/4$. Since $\zeta \geq \rho/\Delta$, the lemma is proved. □

We note an example where this estimate is exact. Suppose $p = q = 2$, $\mathbf{x}^1 = (1,1)$, $\mathbf{x}^2 = (3,1)$, so that $\mathcal{L}$ is the line $y = 1$. Since $\bar{\mathbf{x}} = (2,1)$, we get $\Delta = \sqrt{2}$. The direction numbers $(3,3)$ are

$$2(\mathbf{x},1) \cdot ((2,1) - (1,1)) = 2(\mathbf{x},1) \cdot (1,0) = 2\mathbf{x} \tag{4.12}$$

and $-2\mathbf{x}$. So $\rho = 2\sqrt{2}\,\mathbf{x}$, and we get $\rho^2/4\Delta^2 = \mathbf{x}^2$.

The reader will in fact trivially verify that the estimate $\rho^2/4\Delta^2$ is exact whenever $\mathcal{L}$ is a line. However that is the only such case. Note that (3.6) may be rewritten as

$$\rho = 2\Big[\sum^{j} ((\mathbf{x}-\mathbf{c}) \cdot \mathbf{x}^j)^2\Big]^{1/2}. \tag{4.13}$$

By the Schwarz inequality,

$$\rho \le 2|\mathbf{x}-\mathbf{c}|\Delta. \tag{4.14}$$

equality holding only if $\mathbf{x}-\mathbf{c}$ is parallel to $\mathbf{x}^j = \bar{\mathbf{x}} - \mathbf{x}^j$ for every j, i.e. $\mathcal{L}$ is a line. Otherwise $\rho < 2|\mathbf{x}-\mathbf{c}|\Delta$, i.e. $\rho^2/4\Delta^2 < (\mathbf{x}-\mathbf{c})^2$, and the estimate is short.

Next we show that a small ρ implies that the solution is nearby. We first need some notation.

Identify a maximal linearly independent collection of the a^j, and denote it by $\{a^k\}_{k=1,\ldots,K}$. Denote the corresponding $K \times p$ matrix by a, and put $\Omega = aa^T$. The $K \times K$ matrix Ω is then positive definite. Dencte its smallest eigenvalue by α.

LEMMA C. The distance to the solution satisfies

$$|x - c| \leq p\Delta/2\alpha \quad . \tag{4.15}$$

PROOF. Suppose that x and c correspond to λ and λ^c. Then we may write

$$x - c = \sum_{j=1}^{q} u_j a^j \quad , \tag{4.16}$$

where $u_j = \lambda_j^c - \lambda_j$. We could do this because the u_j sum to zero. Now consider a j for which a^j does not lie in the basis set $\{a^k\}_{k=1,\ldots,K}$. Then a^j is some linear combination of the a^k, which we may substitute into (4.16). After doing this for each such j, we have a representation

$$x - c = \sum_{k=1}^{K} v^k a^k \quad , \tag{4.17}$$

in which the coefficients no longer add to zero; we have not

altered $\bar{x}$. By the choice of the w^k , this representation is unique.

Put

$$\vartheta_k = 2(x - c) \cdot w^k , \quad k = 1, \ldots, K . \tag{4.18}$$

Then, from (4.17),

$$\vartheta_k = 2 \sum_{k'=1}^{K} \nu_{k'} w^k \cdot w^{k'} , \quad k = 1, \ldots, K . \tag{4.19}$$

Now this may be rewritten as

$$\vartheta = 2\Omega\nu , \tag{4.20}$$

where $\vartheta = (\vartheta_1, \ldots, \vartheta_K)$ and $\nu = (\nu_1, \ldots, \nu_K)$. Since Ω^{-1} has the largest eigenvalue α^{-1} , this implies that

$$|\nu| \leq \alpha^{-1} |\vartheta| / 2 . \tag{4.21}$$

Now recall the alternate representation (4.13) for ρ . It follows from that that $|\vartheta| \leq \rho$. Hence (4.21) implies that

$$|\nu| \leq \rho / 2\alpha . \tag{4.22}$$

Now from (4.17), the Schwarz inequality, and the definition (4.2), we get

$$|x - c| \leq |v| \Delta \quad . \tag{4.23}$$

The assertion (4.15) of the lemma now follows from (4.22) and (4.23) . □

We are now ready to prove our convergence theorem.

CONVERGENCE THEOREM. Each move with $x \neq c$ produces a new point x' with

$$|x' - c| < e^{-\alpha^2/2\Delta^4} \times |x - c| \quad . \tag{4.24}$$

where α is the smallest eigenvalue and Δ^2, where Δ is given by (4.2), the trace of the matrix Ω .

PROOF. By (4.15) and Lemma B, the move decreases the squared distance by at least

$$\frac{4\alpha^2 (x - c)^2}{\Delta^2} \times \frac{1}{4\Delta^2} = \frac{\alpha^2}{\Delta^4} (x - c)^2 \quad . \tag{4.25}$$

(4.24) follows immediately from this and the inequality $1 - u < e^{-u}$. □

That the process converges is obvious from this theorem. But it gives no useful information on the speed of convergence. That smallest eigenvalue α could indeed be much smaller than the trace. On the other hand, quite a few extreme estimates went into the derivation of (4.24). How fast the algorithm converges must be tested by experience.

We now turn to the second task of this section. the analysis of the tests involving the value. We have first to quote a variant of a principle due to C. E. Lemke.

LEMKE'S CLOSEST-POINT PRINCIPLE FOR EQUALITY CONSTRAINTS [Lemke [1], 1961]. Consider the set B of all linear combinations $b = \sum \mu_j b_j$ of the vectors b_j. There are two possibilities. It may be that $z \in B$, in which case $z \cdot \gamma = 0$ for all γ satisfying (1.2). Or else $z \notin B$. In this case there is a point $b(z)$ of B closest to z. Put

$$v = |z - b(z)|, \quad \gamma = (1/v)(z - b(z)) . \tag{4.26}$$

Then γ solves the problem (1.1)-(1.2), and the value is v *).

---

*) Lemke's formulation had $b_j \cdot \gamma \leq 0$ instead of the equalities we have at (1.2). His principle then requires the μ_j to be nonnegative; everything else is the same.

The proof of this principle is trivial; one simply writes out the formula for the squared distance in terms of u and differentiates. But it does not seem to have been noticed before Lemke.

LEMMA D. For the γ of (3.2), always

$$z \cdot \gamma = |h| \geq v . \tag{4.27}$$

PROOF. As to the equality, first observe that

$$x \cdot h = z \cdot x - x^2/x^2 = 1 - 1 = 0 , \tag{4.28}$$

a calculation similar to (2.6). Hence

$$z \cdot v = (1/|h|)\, z \cdot h = (1/|h|)\,(z - x/x^2) \cdot h = (1/|h|)\, h^2 = |h| . \tag{4.29}$$

As to the inequality, it is evident that the linear manifold $\mathcal{L}$ is a (proper) subset of Lemke's cone B, and so $|h|$ a distance $|z - b|$ with $b \in B$. The inequality thus follows from Lemke's principle. □

So (1.3) in fact holds always without the ϵ, and the test $|h| \leq \epsilon$ in the algorithm is correct.

§ 5. On minimization in a class of quadratic forms

The idea of this section derives from Lemke's principle, which we now write out somewhat more explicitly. For $u \in R^q$ put

$$\vartheta_i(u) = \sum_{j=1}^{q} b_{ji} u_j \ , \quad i = 1, \ldots, p \ , \tag{5.1}$$

and then

$$\Im(u) = \vartheta(u) \cdot \vartheta(u) \ , \tag{5.2}$$

the dot product being in R^p. The principle says that the problem of steepest ascent is equivalent to the problem of minimizing the quadratic form $\Im(u)$.

Now we may consider the transformation $u \to \vartheta$ of (5.1) as mapping the "base space" of the "Lagrange multipliers" onto the cone B in the "image space" R^p of the x. The difficulty in trying to work in the base space is that there is no intrinsic connections between the local (metric) geometries. Steepest ascents, or even good ascents, do not correspond. For instance,

in the original steepest-ascent problem (1.1)-(1.2), multiplying one of the b_j's through by, say, 2, does not change the problem at all, but it considerably affects the geometry of the base space. It was only after considerable efforts to make schemes based on the base space work that the author came to the idea of the linear submanifold $\mathcal{L}$ of B, as a way of getting at "steepest feasible ascent" in the image space directly. There are intrinsic relationships between the natural metric x-geometry and the metric λ-geometry of the parameters, as we showed above in Lemmas A-C. That is why our algorithm should work well for the steepest-ascent problem.

And now we come to out point. For the same reasons, it should work well on the problem of minimizing any quadratic form given as the sum of squares of linear forms, as is the $\Theta(u)$ of (5.1)-(5.2), in all of R^q without boundaries, regardless of provenance. All one has to do is treat it as a steepest-ascent problem, apply our algorithm to obtain a sufficiently good set $\{\lambda_j\}$, and then put $u_j = \beta_j \lambda_j$, $j = 1, \ldots, q$, the β_j being given by (2.1).

So we have turned Lemke's principle around.

LITERATURE

[1] C. E. Lemke, "The constrained gradient method of linear programming", Journal of the Society of Industrial and Applied Mathematics 9 (1961) 1-17.

ENCLOSURE B
A VARIANT OF KARMARKAR'S LINEAR PROGRAMMING ALGORITHM NOT REQUIRING PROJECTION ONTO THE NULL SPACE

BY JOHN M. DANSKIN

ABSTRACT

Recently, N. Karmarkar proposed to solve a linear program by mapping the problem projectively into a simplex, using further projective transformations to keep the current iteration point at the barycenter of the simplex. He finds the direction of steepest descent from that barycenter by projecting his objective vector onto the feasible (null) space. This method, which requires the inversion of a matrix, has led to severe difficulties of speed and accuracy in inverting the matrix, of finding an initial point, and questions as to whether convergence has in fact been proved.

The present paper uses Karmarkar's simplex, and his projective transformations, but is otherwise very different from his. The method, which involves bringing down a number of "deficits" in a "balanced" way, requires no projection onto any null space, no matrix inversion, no operations affecting the sparsity of the matrix. It has no problems with accuracy.

There is a classical, real-number, proof of convergence, not involving bounds based on the number of bits required to state the problem. A FORTRAN program is being prepared for testing.

We do not offer a complexity-type estimate of the speed of the algorithm, because we do not have one. We do not know how many steps the "inside" steepest-ascent algorithm requires to achieve given accuracy, because the convergence estimate for that algorithm, given at (4.24) in [1], contains quantities very difficult to estimate. What we do have is a complete proof of the convergence of the whole process. The speed of this algorithm can only be tested by experience.

§ 1. Introduction

This paper presents a new version of Karmarkar's approach [2, 3] to the solution of a linear program by mapping the problem into a simplex and working thenceforth in the simplex.

Our method has none of the feasibility or proof-of-convergence problems that have arisen in Karmarkar's approach. Furthermore, it has no projection onto the null space and no matrix inversion.

It works by setting up a system of "deficits" in the satisfaction of the conditions for a solution of an LP, including the feasibility conditions. We introduce new variables, analogous to the slack variables of the simplex method but not really slack variables, so as to make the deficits all equal and positive at the outset. Working then in Karmarkar's simplex, we find a direction in which the deficits all go down by guaranteed amounts, in such a way that the ratio of the largest to the smallest deficit never exceeds 5/3 .

We find that direction by a new geometric method for finding approximate steepest-ascent subject to equality constraints, developed [1] for the present purpose. The constraints have to be met only very loosely: see (4.9) and (4.10). The sparseness of the matrix is not affected. While some precision is important in setting up the linear manifold $\mathcal{L}$ on which the method of [1] works, there is no accumulation of errors. These are in fact self-correcting, as we show in § 4.

At the time of this writing, the method has not been tested. A FORTRAN program is in an advanced state of preparation, and should be working by July. While the theory presented in this paper is complete and ready for refereeing, it will not be submitted for publication until the method has been tested against commercial fast simplex methods like Tomlin's WHIZZARD, under proper conditions, and we have comparisons to report. Meanwhile, this paper is to be regarded as a private working document, for the author's use in preparing his program, as documentation in proposals, and for those of his colleagues who wish to read it.

The organization is as follows. §2 states the problem, defines the deficits, and states our objectives. § 3 explains our approach to the simplex, in rather more detail and with an attitude rather different from Karmarkar's. He, for instance, has only one simplex, and one "potential function" . We have infinitely many simplexes, and infinitely many "surrogate functions" . The main result of that section is the estimate (3.26) for the current deficit, given in the original space: Karmarkar has given no such estimate. That estimate has on its right hand side two quantities, an μ that we have to prove less than unity, and a "geometric mean" of the coordinates of the current position point in the original space, which we have to prove bounded.

In §4 we show how to move so as to get an improvement while preserving the 5/3 ratio between the largest and smallest deficits, a central aspect of our approach. In §5 we show that the move of §4 yields a guaranteed gain, and estimate μ at (5.19) . In § 6 we prove, by contradiction, that the "geometric mean" appearing in (3.26) is bounded, and thus, with the results of §5 on μ, complete the convergence proof. The final §7 outlines our routine.

§ 2. Statement of the problem

We consider the LP

$$\underline{\text{Maximize}} \quad C \cdot X = \sum_{i=1}^{p} C_i X_i \quad \underline{\text{subject to}} \quad \sum_{i=1}^{p} A_{ij} X_i \leq B_j, \; j = 1, \ldots, q \tag{2.1}$$

and its dual

$$\underline{\text{Minimize}} \quad B \cdot Y = \sum_{j=1}^{q} B_j Y_j \quad \underline{\text{subject to}} \quad \sum_{j=1}^{q} A_{ij} Y_j \geq C_i, \; i = 1, \ldots, p, \tag{2.2}$$

in which $X \geq 0$, $Y \geq 0$. The primal variable X is said to be feasible if it satisfies the inequalities in (2.1), and the dual variable Y feasible if it satisfies the inequalities in (2.2). It is trivial that if X and Y are both feasible, then $C \cdot X \leq B \cdot Y$; and a necessary and sufficient condition that the pair (X, Y) solve the problems (2.1)-(2.2) is that $C \cdot X = B \cdot Y$.

The standard approach to (2.1)-(2.2) is, first, to find a feasible X. This, the "first phase", requires the solution of an auxiliary linear program. Then, in the "second phase", keeping X always feasible, one seeks to increase $C \cdot X$. In the simplex method, one arrives eventually at a "basic optimal solution", in terms of a "basis". The test for optimality is

made in terms of this basis; and the solution Y to the dual problem is obtained automatically from the operations needed to solve the primal.

Our approach is wholly different. We never have a feasible X or Y. Rather, we have at any time some nonnegative pair (X, Y), and associated with that pair a collection of "deficits", amounts by which the inequalities in (2.1) and (2.2), and the inequality $C \cdot X \geq B \cdot Y$, are violated. If none of them is violated, then we have, by what was said above, a classical solution. If several of them are violated, we seek to bring the violations, the deficits, down. For us, a solution is a pair (X, Y) for which none of the deficits exceeds some prescribed small positive number. And we do not attempt to provide a basis.

A basic difficulty with any scheme for reducing deficits is that a move intended to reduce the deficits at hand may introduce new ones. We get around this by arranging things so that everything is always in deficit. We introduce new nonnegative variables U_i, $i = 1, \ldots, p$; V_j, $j = 1, \ldots, q$; and W, and put

$$\Delta_i = -\sum_{j=1}^{q} A_{ij} Y_j + U_i + C_i , \quad i = 1, \ldots, p , \tag{2.3}$$

$$\Delta_j = \sum_{i=1}^{p} A_{ij} X_i + V_j - B_j , \quad j = 1, \ldots, q , \tag{2.4}$$

$$\Delta = -\sum_{i=1}^{p} C_i X_i + \sum_{j=1}^{q} B_j Y_j + W . \tag{2.5}$$

At the outset we choose the U_i , V_j , and W are all positive and equal. We from then on move in such a way that always

$$\Delta_i , \Delta_j \in [\tfrac{3\Delta}{4} , \tfrac{5\Delta}{4}] . \tag{2.6}$$

That this can be accomplished, while driving Δ (generally) down, is nontrivial, and a principal element of our method. See §4 .

So we are now operating in the space of the (X, Y, U, V, W), in the nonnegative orthant of $R^{2p+2q+1}$. Our objective, given a positive ϵ , is to find a set (X, Y, U, V, W) for which (2.6) is satisfied and

$$0 < \Delta < \epsilon . \tag{2.7}$$

We are now ready to transfer the problem to the Karmarkar simplex.

δ_j^0/z , and δ^0/z , where

$$\delta_i^0(\xi) = -\sum_{j=1}^{q} A_{ij} Y_j^0 y_j + U_i^0 u_i + C_i z \ , \ i = 1, \ldots, p : \qquad (3.5)$$

$$\delta_j^0(\xi) = \sum_{i=1}^{p} A_{ij} X_i^0 x_i + V_j^0 v_j - B_j z \ , \ j = 1, \ldots, q ; \qquad (3.6)$$

$$\delta^0(\pi) = -\sum_{i=1}^{p} C_i X_i^0 x_i + \sum_{j=1}^{q} B_j Y_j^0 y_j + W^0 w \ . \qquad (3.7)$$

For ξ strictly interior to Σ , we put

$$\varphi_i^0(\xi) = \delta_i^0(\xi)/\sigma(\xi) \ , \ i = 1, \ldots, p : \ \varphi_j^0(\xi) = \delta_j^0(\xi)/\sigma(\xi) \ , \ j = 1, \ldots, q ; \qquad (3.8)$$

$$\varphi^0(\pi) = \delta^0(\xi)/\sigma(\pi) \ ,$$

where $\sigma(\xi)$ is the geometric mean function

$$\sigma(\xi) = (\xi_1 \cdot \ldots \cdot \xi_{m+1})^{1/(m+1)} \ . \qquad (3.9)$$

The functions φ_i^0 , φ_j^0 , and φ^0 are then homogeneous of degree zero. Karmarkar calls them (their logarithms) "potential functions" . I prefer to call them "surrogate functions" .

§ 3. The problem on the simplex

Put $m = 2p + 2q + 1$ and write $\Xi = (X, Y, U, V, W)$. Choose a $\Xi^0 \in R^m$ with $\Xi^0_k > 0$, $k = 1, \ldots, m$, and so that all the Δ_i, Δ_j, and Δ of (2.3)-(2.5) are equal. For any $\Xi \in R^m_+$ put

$$\xi_{m+1} = z = 1/(1 + \sum_{k=1}^{m} \Xi_k/\Xi^0_k) , \tag{3.1}$$

and then

$$\xi_k = z\,\Xi_k/\Xi^0_k , \quad k = 1, \ldots, m . \tag{3.2}$$

These formulas define a transformation T^0, called the Karmarkar transformation, of R^m_+ into the unit simplex

$$\Sigma: \quad \xi_1 + \ldots + \xi_{m+1} = 1 , \ \xi_k \geq 0 , \quad k = 1, \ldots, m+1 \tag{3.3}$$

in R^{m+1}. Its inverse, defined for $\xi_{m+1} = z > 0$, is

$$(T^0)^{-1}: \qquad \Xi_k = \xi_k\,\Xi^0_k/\xi_{m+1} , \quad k = 1, \ldots, m . \tag{3.4}$$

We will use the obvious notation $\xi = (x, y, u, v, w, z)$. The functions Δ_i, Δ_j, and Δ transform under T^0 into the functions δ^0_i/z,

The transformation T^0 has mapped the starting point $\Xi^0 \in R^m_+$ into the barycenter $\bar{\xi}$ of Σ, all of whose components are $1/(m+1)$. Moves, in Karmarkar's scheme, are made as follows. One distinguishes some $\xi^\#$ interior to Σ, in some sense better than $\bar{\xi}$. One then applies the transformation

$$t^\#: \qquad \xi'_k = \frac{\xi_k/\xi^\#_k}{\sum \xi_k/\xi^\#_k} \quad , \quad k = 1, \ldots, m+1 , \qquad (3.10)$$

which maps $\xi^\#$ into $\bar{\xi}$. Karmarkar thinks of one simplex Σ, with one potential function (really an equivalence class) defined on it. I prefer to think of replicas of Σ, so that the original simplex, into which T^0 maps, is Σ^0, and the first of the transformations $t^\#$ maps Σ^0 onto Σ^1. Put $\Xi^1 = (T^0)^{-1}(\xi^\#)$. Then the product transformation $T^1 = T^0 t^\#$, mapping R^m_+ into Σ^1, is given by the formulas (3.1)-(3.2) with Ξ^0 replaced by Ξ^1, and it maps Ξ^1 into the barycenter of Σ^1. In general suppose that $N \geq 1$ and that we have defined a transformation T^{N-1} mapping R^m_+ into a replica Σ^{N-1} of Σ, as a mapping of the type (3.1)-(3.2) with Ξ^0 replaced by a $\Xi^{N-1} \in R^m$ having all its components positive. Suppose a distinguished $\xi^\#$ interior to Σ^{N-1} is at hand, and that the inverse of $\xi^\#$ under T^{N-1} is Ξ^N. Then Ξ^N has all its components positive. Now use (3.10) to define a mapping of Σ^{N-1} onto a further replica Σ^N of Σ, and then put $T^N = T^{N-1} t^\#$. T^N then maps R^m_+ into Σ^N, and

it is given by (3.1)-(3.2) with Ξ^0 replaced by Ξ^N. This completes the inductive definition of the T^N, except for the precise specification of the distinguished point $\xi^\#$, which we shall give below in § 7.

We will need an identity, based on the above definitions. We defined Ξ^N above to be the inverse of $\xi^\#$ under T^{N-1}. Since the inverse of T^{N-1} is gotten from (3.4) by replacing the superscript 0 at both of its appearances by $N-1$, this means that

$$\Xi_k^N = \xi_k^\# \Xi_k^{N-1} / \xi_{m+1}^\# \quad , \quad k = 1, \ldots, m \, . \tag{3.11}$$

Now suppose $\xi \in \Sigma^{N-1}$ and define $\xi' \in \Sigma^N$ by (3.10). On writing $\sigma = \sum \xi_k / \xi_k^\#$, we get

$$\Xi_k^N \xi_k' = (\xi_k^\# \Xi_k^{N-1} / \xi_{m+1}^\#) \times (\xi_k / \sigma \xi_k^\#) = \Xi_k^{N-1} \xi_k / \sigma \, \xi_{m+1}^\# \, ,$$

$$k = 1, \ldots, m \, . \tag{3.12}$$

Also evidently

$$\varrho(\xi') = \varrho(\xi) / \sigma \varrho(\xi^\#) \, . \tag{3.13}$$

It follows that

$$\Xi_k^N \xi'_k = \Xi_k^{N-1} \xi_k \times \frac{\varphi(\xi')}{\varphi(\xi)} \times \frac{\varphi(\xi^{\#})}{\xi^{\#}_{m+1}} \quad , \quad k = 1, \ldots, m \quad . \qquad (3.14)$$

This is the required identity.

We now define δ_i^N/z , δ_j^N/z , and δ^N/z to be the transforms, in Σ^N , of Δ_i, Δ_j , and Δ under T^N . The functions δ_i^N , δ_j^N , and δ^N are given by (3.5)-(3.7) with the superscripts 0 replaced by N. And we define the surrogate functions φ_i^N , φ_j^N , and φ^N by making the same replacements in (3.8) .

We now apply the identity (3.14) to (3.8), with 0 replaced by N. It gives

$$\delta^N(\xi') = \delta^{N-1}(\xi) \times \frac{\varphi(\xi')}{\varphi(\xi)} \times \frac{\varphi(\xi^{\#})}{\xi^{\#}_{m+1}} \quad . \qquad (3.15)$$

From the definition (3.8) with 0 replaced by N - 1 , we get

$$\varphi^N(\xi') = \frac{\varphi(\xi^{\#})}{\xi^{\#}_{m+1}} \times \varphi^{N-1}(\xi) \quad . \qquad (3.16)$$

This identity, which Karmarkar never stated explicitly because he had only one potential function and one simplex (the ratio on the right appears as the rather vaguely explained additive term

in his definition of the potential function, additive because he works with the logarithm) , is the fundamental identity of his whole construction, and the reason why it works. It also explains why he needs the surrogate function φ^N and not some other function, say δ^N .

Observe the following consequence. Suppose that $\xi, \eta \in \Xi^{N-1}$, $\xi' = t^{\#}\xi$, $\eta' = t^{\#}\eta$. Then

$$\frac{\varphi^N(\xi')}{\varphi^N(\eta')} = \frac{\varphi^{N-1}(\xi)}{\varphi^{N-1}(\eta)} \quad . \tag{3.17}$$

This is Karmarkar's "invariant projective cross-ratio", to which he alludes but never makes explicit. It means that one can track the progress of the algorithm in any of the replicas of Σ , in particular Σ^0 , which we are about to do. We note that (3.17) holds for the φ_i^N and φ_j^N as well.

In what follows we suppose that at each stage N=0.1,... we are able to find a distinguished $\xi^{\#} \in \Sigma^N$ such that

$$\varphi^N(\xi^{\#}) < w\varphi^N(\bar{\xi}) \quad , \tag{3.18}$$

$w \in (0,1)$ being independent of N . Using this and the apparatus developed above, we find at (3.26) an estimate for $\Delta(\Xi^N)$. The

proof is close to Karmarkar's (incomplete) proof of convergence for the case when the value is zero in his early manuscript [2].

We said we would track the progress of the algorithm in Σ^0. So we put

$$\xi^N = T^0 \Xi^N , \quad N = 0, 1, \ldots . \tag{3.19}$$

The ξ^N are now in Σ^0, with $\xi^0 = \bar{\xi}$.

Now (3.18) and the invariant cross-ratio principle (3.17) imply that

$$\varphi^{N-1}((t^{\#}_{N-1})^{-1} \xi^{\#}) < w \varphi^{N-1}(t^{\#}_{N-1})^{-1} \bar{\xi}) , \tag{3.20}$$

where $t^{\#}_{N-1}$ denotes the mapping $t^{\#}$ of (3.10) when it is from Σ^{N-1} onto Σ^N. That is, the inequality (3.18) on Σ^N implies the same inequality on Σ^{N-1}, but applied to the predecessors. Walking this back to Σ^0 and recalling the definition of Ξ^{N+1} as the inverse of $\xi^{\#}$ under T^N, we get

$$\varphi^0(\xi^{N+1}) < w \varphi^0(\xi^N) . \tag{3.21}$$

Thus we have tracked the action back to Σ^0. Now concatenate

the inequalities (3.21), with N replaced by $0,1,\ldots,N-1$. We get

$$\varphi^0(\xi^N) < \omega^N \varphi^0(\xi^0) \ , \quad N = 0,1,\ldots \quad . \tag{3.22}$$

We now have to determine what this means relative to the original problem in R_+^m : this is what Karmarkar did not do in [2] .

We recall that $\xi^0 = \bar{\xi}$. From (3.7) and (3.8) ,

$$\varphi^0(\bar{\xi}) = -\sum_{i=1}^{p} C_i X_i^0 + \sum_{j=1}^{\bar{n}} B_j Y_j^0 + W^0 \ , \tag{3.23}$$

which is equal to the original deficit $\Delta(\Xi^0)$. From the definitions of δ^0, Ξ^N and ξ^N, we have

$$\Delta(\Xi^N) = \delta^0(\xi^N)/z^N \ , \tag{3.24}$$

in which z^N is the last component of ξ^N . From this and (3.23), we see that (3.22) translates into

$$\Delta(\Xi^N) < \omega^N \times \Delta(\Xi^0) \times \varrho(z^N)/z^N \ . \tag{3.25}$$

We calculate the last factor in (3.25) by substituting (3.2) into the definition (3.9). This gives us

$$\Delta(\Xi^N) < \frac{\omega^N \Delta(\Xi^0)\,[\Xi_1^N \cdot \ldots \cdot \Xi_m^N)^{1/(m+1)}}{[\Xi_1^0 \cdot \ldots \cdot \Xi_m^0]^{1/(m+1)}} \,. \tag{3.26}$$

This is our desired estimate of the deficit at stage N .

In order to pass from (3.26) to a proof of convergence, we have to solve two problems. The first is that of finding the distinguished $\xi^{\#}$ in (3.18) , while respecting the condition

$$\delta_i^N \,,\ \delta_j^N \in [\,\frac{3\delta^N}{4} \,,\ \frac{5\delta^N}{4}\,] \tag{3.27}$$

corresponding to (2.6) . That we do in §4 below. That that moves yields a guaranteed $\omega \in (0,1)$ is proved in §5 : see (5.19) . The denominator in (3.26) gives us no trouble: we could have chosen all the initial Ξ_k^0 to exceed unity. So the second, and final, problem is that of finding conditions under which the "geometric mean" term in the numerator is bounded. We prove in §6 that this is so provided the set of solutions of the original LP problem (2.1)-(2.2) is bounded.

§4. The balanced reduction of deficits

We suppose ourselves at the barycenter at stage N, and that the δ_i^N, δ_j^N, δ satisfy the ratio condition (3.28). Our objective is to move in such a way as to guarantee a decrease in δ and to preserve (3.27).

The situation is substantially more complicated than it is in Karmarkar's algorithm. He knows that there is a solution somewhere in the simplex, and not as far as one unit away. Hence there is at least a certain rate of decrease in the direction towards the solution, and therefore at least that much in the direction of steepest descent.

What we need is more than just the existence of a solution somewhere in the simplex. We need not only a pull downward, but a pull towards correcting the ratios when they begin to diverge. What we need is to know the existence of points which are well, but not too well, down, and have all the deficits equal.

To do this, we go to the point $\Xi^N = (X^N, Y^N, U^N, V^N, W^N) \in R_+^m$ at hand, find the largest deficit, and adjust the variables U_i, V_j, and W corresponding to the other deficits upwards, so as to make the deficits all equal. This gives us a point Ξ^*, whose image

in Σ^N we denote by ξ^* . Now there are two possibilities. We consider the easy one first.

That is the case when

$$\delta^N(\xi^*) \leq \delta(\bar{\xi})/2 \ . \tag{4.1}$$

In this case we move towards ξ^* , either to the (approximate) minimum of $\varphi^N(\xi)$ or to ξ^* , whichever comes first. In this case the ratios will be brought back towards, perhaps all the way to, unity. And there will be a guaranteed proportionate decrease: see § 5 .

The more difficult case is

$$\delta^N(\xi^*) > \delta^N(\bar{\xi}) / 2 \ . \tag{4.2}$$

We now drop the superscript N and write $\bar{\delta}_i$, $\bar{\delta}_j$, $\bar{\delta}$ for $\delta_i(\bar{\xi})$, $\delta_j(\bar{\xi})$, $\bar{\delta}(\bar{\xi})$. Now , on the assumption that the original problem has a solution, there is a point ξ^{**} in Σ with δ_i, δ_j , and δ all zero. Consider the line $[\xi^*, \xi^{**}]$. There is obviously on that line a point ξ^{***} satisfying

$$\delta_i(\xi^{***}) = \delta_j(\xi^{***}) = \delta(\xi^{***}) = \bar{\delta}/2 \tag{4.3}$$

for all i, j . It will be this unknown point we will try to aim at, not the solution.

Consider the line segment $\ell = [\bar{\xi}, \xi^{***}]$. Denote the length of ℓ by d , and derivatives along it by primes. Then

$$\bar{\delta}_i - \bar{\delta} + d(\delta_i' - \delta') = 0 \ . \tag{4.4}$$

Also,

$$d\delta' = -\bar{\delta}/2 \ . \tag{4.5}$$

On eliminating the d we get

$$\delta_i' = \delta'(2\bar{\delta}_i/\bar{\delta}) - 1) \ . \tag{4.6}$$

Similarly we derive

$$\delta_j' = \delta'(2\bar{\delta}_j/\bar{\delta} - 1) \ . \tag{4.7}$$

So: we have proved that in the case (4.2) there exists a direction respecting the p constraints (4.6), the q constraints

(4.7), and the simplex equation in (3.3), and at the same time bringing down δ at a rate exceeding $\bar{\delta}/2$.

What we will do, knowing the above fact, is to set in motion the steepest-ascent routine of [1], asking for the steepest ascent of $-\delta$ which respects the constraints (4.6), (4.7), and the simplex constraint. This routine, being iterative and approximate, will never, except by remote chance, ever achieve any of those equations exactly. But we do not need exactness, except in the simplex constraint. At each step of the iteration, we will "plaster" the current direction vector γ' onto the simplex by subtracting, from each component, the average of the components, and then renormalizing. We then test the inequality

$$-\delta' > |h|/2 \ , \tag{4.8}$$

in which δ' is the rate corresponding to the plastered γ, and h is given by (3.1) in [1]. This will assure that the plastered vector is yielding more than half the maximum rate of descent subject to the constraints; we explain this at the beginning of §5. If (4.8) passes, we test the inequalities

$$\delta_i'/\delta' \in [2\bar{\delta}_i/\bar{\delta} - 5/4, 2\bar{\delta}_i/\bar{\delta} - 3/4] \tag{4.9}$$

and

$$\delta_j'/\delta' \in [\,2\bar{\delta}_j/\bar{\delta} - 5/4\,,\ 2\bar{\delta}_j/\bar{\delta} - 3/4\,]\ . \tag{4.10}$$

We will accept the candidate plastered direction if all these inequalities are satisfied. Suppose an i is accepted with

$$\delta_i'/\delta' = 2\bar{\delta}_i/\bar{\delta} - \alpha\ , \tag{4.11}$$

where $\alpha \in [3/4, 5/4]$. Suppose one moves in the candidate direction, with $\delta' = -s$. Then $\delta_i' = -s(2\bar{\delta}_i/\bar{\delta} - \alpha$, so that at a given distance d

$$\delta_i(d) = \delta_i - sd(2\bar{\delta}_i/\bar{\delta} - \alpha)\ . \tag{4.12}$$

Choose d^* (which might be greater than the distance to the boundary) so that $sd^* = \bar{\delta}/2$. Then $\delta_i(d^*) = \alpha\bar{\delta}/2$, i.e.

$$\delta_i(d^*)/\delta(d^*) = \alpha\ . \tag{4.13}$$

Since the ratio $\delta_i(d)/\delta(d)$ is monotone, and since it lies in $[3/4, 5/4]$ at both $d = 0$ and $d = d^*$, it therefore lines on that interval for any $d \in [0, d^*]$. The same obviously goes for any j. It follows that the ratio condition (3.27) will be respected by any move that does not reduce δ by more than half. □

§ 5. Estimation of the improvement on a move

The purpose of this section is to estimate the guaranteed proportionate reduction in $\varphi^N(\xi)$ on a move of one of the two types described in §4, and to discuss step size.

We have first to discuss the effect of the "plastering" of the direction vector to the simplex, and the meaning of the test (4.8), both occurring in the "difficult" case (4.2). As to the plastering, for any positive ϵ the routine of [1] will eventually produce a γ' with $|s| < \epsilon$, where

$$s = \sum_{k=1}^{m+1} \gamma'_k \quad . \tag{5.1}$$

Subtracting off $s/(m+1)$ from each γ'_k gives us a vector γ'' satisfying $|\gamma'' - \gamma'| < \epsilon/(m+1)^{1/2}$. And the length of γ'' differs from unity by less than $3\epsilon^2/2(m+1)$. So the plastered γ differs from γ' by only about $\epsilon/(m+1)^{1/2}$. Hence the additional conditions (4.9)-(4.10) will certainly eventually be met. The operation of plastering is therefore almost free of charge.

We now come to the question of the meaning of the test (4.8) . We need to have a relationship between the rate of descent λ (we mean $\lambda > 0$) provided by the plastered γ and the current value of $\bar{\delta} = \delta^N(\bar{\xi})$. Denote by λ^0 the exact steepest rate of descent for the constrained problem. We do this in three steps. We first relate λ to the λ' provided by the unplastered vector γ' . This is accomplished by (4.8), which tells us that $\lambda > \lambda'/2$. We then relate λ to λ^0 , and finally λ^0 to $\bar{\delta}$.

As to the second relation, it is a characteristic of the steepest ascent algorithm that always $\lambda' \geq \lambda^0$. This we proved at Lemma D in §4 of [1] . What happens in that algorithm is that the current value comes down as the deviations from the constraints decrease.

As to the final relation, recall that in this case there exists a ξ^{***} satisfying (4.3) . δ decreases by exactly $\bar{\delta}/2$ on the segment $[\bar{\xi}, \xi^{***}]$. And the length of that segment is less than

$$R = \sqrt{m/(m+1)} \quad , \tag{5.2}$$

the radius of Karmarkar's circumscribed sphere. So the rate of descent on that segment exceeds $\bar{\delta}/2R$ Now the direction of that segment satisfies by definition the constraints of the steepest-ascent problem. It follows that $\lambda^0 > \bar{\delta}/2R$. We therefore have

$$\lambda > \bar{\delta}/4R \ , \tag{5.3}$$

which is the relation we need.

That explains the test (4.8), in the case (4.2) . In the other case (4.1), the direction we have chosen, the one pointed straight towards ξ^* , takes δ down by at least $\bar{\delta}/2$ in a distance less than R . The λ for that direction therefore exceeds $\bar{\delta}/2R$. Hence (5.3) holds in that case as well. □

The rest of this argument is simply a very-much-reworked, simplified, and in some respects amplified, version of Karmarkar's derivation of the corresponding estimate in [2] . In particular we go 1/4 of the way across the inscribed sphere.

We have first to write out the simple properties of the geometric mean function $g(\xi)$ of (3.9) in the direction of the move.

Put

$$G(t) = \rho(\bar{\xi} + \gamma t) \ . \tag{5.4}$$

Then

$$\dot{G}(t) = \frac{G(t)}{m+1} \, h(t) \ , \tag{5.5}$$

where

$$h(t) = \sum_{k=1}^{m+1} \frac{\gamma_k}{\xi_k(t)} \ . \tag{5.6}$$

We get

$$\dot{h}(t) = - \, k(t) \ , \tag{5.7}$$

where

$$\sum_{k=1}^{m+1} \frac{\gamma_k^2}{[\xi_k(t)]^2} \ . \tag{5.8}$$

So $h(0) = 0$, and $h(t)$ is strictly decreasing. Next,

$$\ddot{G}(t) = \frac{G(t)}{m+1} \left[\frac{(h(t))^2}{m+1} - k(t) \right] \ . \tag{5.9}$$

By the Schwarz inequality, the quantity in brackets is nonpositive, and it can be zero only if $\gamma_k = \omega\xi_k(t)$ for all $k = 1, \ldots, m+1$, ω being some proportionality constant. Since the γ_k sum to zero and the $\xi_k(t)$ to unity, we would then have $\omega = 0$, so that all the γ_k are zero, impossible because γ is a unit vector. Hence the quantity in brackets is in fact strictly negative, so that G(t) is strictly concave.

Put

$$r = \sqrt{1/m(m+1)} \quad . \tag{5.10}$$

This is the radius of Karmarkar's inscribed sphere.

We need an estimate for h(t), on the interval $[0, r/4]$. We assume that $m > 2$. Then

$$\xi_k(t) > \frac{1}{m+1} - \frac{1}{4\sqrt{m(m+1)}} > \frac{2}{3(m+1)} \quad . \tag{5.11}$$

Hence

$$k(t) < \frac{9}{4}(m+1)^2 \sum_{k=1}^{m+1} \gamma_k^2 = \frac{9}{4}(m+1)^2 \quad . \tag{5.12}$$

It follows that

$$0 < -h(t) < \frac{9r}{16}(m+1)^2 \tag{5.13}$$

on $[0, r/4]$. This is the desired estimate.

Now put

$$\Phi(t) = \frac{\delta(\xi(t))}{G(t)} \quad . \tag{5.14}$$

This is the objective function $\varphi(\xi)$, taken along the line. Then

$$\dot{\Phi}(t) = -\frac{1}{G(t)}\left[\lambda + \frac{\delta(\xi(t))}{m+1}\dot{n}(t)\right] . \tag{5.15}$$

Using (5.3) and (5.13), we estimate the second term in brackets:

$$0 < -\frac{\delta(\xi(t))}{m+1}h(t) < -\frac{\bar{\delta}}{m+1}h(t) < \frac{4R\lambda}{m+1} \times \frac{9r}{16}(m+1)^2 = \frac{9\lambda}{16} . \tag{5.16}$$

Hence

$$\dot{\Phi}(t) < -\frac{7\lambda}{16} \times \frac{1}{G(t)} \quad . \tag{5.17}$$

Putting this together with the definition (5.14) and using (5.3) once again, we get

$$\frac{d}{dt}[\log \Phi(t)] < -\frac{7\lambda}{16} \times \frac{1}{4R\lambda} = -\frac{7}{64R} . \qquad (5.18)$$

On integrating this across $[0, r/4]$, we get

$$\Phi(r/4) < \Phi(0) e^{-7/256m} . \qquad (5.19)$$

Hence we would have (3.18) and hence (3.26), with $\omega = e^{-7/256m}$, if we were to choose $\xi^{\#} = \xi(r/4)$. We have thus achieved the desired guaranteed proportional reduction in the surrogate function φ. □

Move length. It would be absurd to make the above choice of $\xi^{\#}$ in computational practice. Karmarkar in [2], and I here, chose it only to obtain the estimate (5.13) for h(t) and hence the guarantee (5.16). In fact the function $\Phi(t)$ of (5.14), as the quotient of a linear function and a strictly concave function, has a unique minimum on the line, which can be many small sphere radii out. It is a trivial exercise, requiring very little computation compared to that required in case (4.2) to obtain the direction, to obtain an approximate minimum using a few steps of a halving process. It is by such a halving process that I propose to choose $\xi^{\#}$ *).

*) [Footnote next page]

[Footnote to previous page]

---

*) Karmarkar had not, until I called his attention to it in October 1984, yet noticed the unique minimum property of $\Phi(t)$. Perhaps his use of the logarithmic version obscured that property. He had been using a "three-point test" .

---

§6. Convergence

The principal objective of this section is to prove the following assertion.

LEMMA. If the set of classical solutions of the original LP problem (2.1)-(2.2) is nonempty and bounded, then the "geometric mean" term

$$(\Xi_1^N \cdot \ldots \cdot \Xi_m^N)^{1/(m+1)} \tag{6.1}$$

appearing in the estimate (3.26) is bounded.

PROOF. We will prove that if that term is unbounded, then there exists an unbounded solution set for the original LP.

Suppose then that the quantity (6.1) tends to infinity on a subsequence of the N. Then it will eventually exceed unity and therefore be less than the true geometric mean $(\Xi_1^N \cdot \ldots \cdot \Xi_m^N)^{1/m}$. Since trivially

$$(\Xi_1^N \cdot \ldots \cdot \Xi_m^N)^{1/m} \leq \frac{(\Xi_1^N + \ldots + \Xi_m^N)}{m}, \tag{6.2}$$

we may replace the factor (6.1) in (3.26) by S/m, where we have written $S = \Xi_1^N + \ldots + \Xi_m^N$ Recalling the definition (2.5), we then have

$$0 < -\sum_{i=1}^{p} C_i X_i'^N + \sum_{j=1}^{q} B_j Y_j'^N + W'^N < \Lambda \varpi^N \Delta(\Xi^0)/m , \qquad (6.3)$$

where we have written $\Xi'^N = \Xi^N/S$ and $\Lambda = (\Xi_1^0 \cdot \ldots \cdot \Xi_m^0)^{-1/(m+1)}$. Since always $\Lambda_i, \Lambda_j \leq (5/4)\Delta$, we get similar estimates from (2.3) and (2.4), with an additional factor 5/4 on the right hand side. Because $\varpi \in (0,1)$ and is constant, and because everything else on the right hand side of (6.3) is constant, the term between inequality signs in (6.3) tends to zero. This is so also for the estimates obtained from (2.3) and (2.4). Since Ξ'^N is on the unit simplex in R_+^m, a subsequence converges to a limit Ξ^*, also on the simplex. We thus find that

$$\sum_{j=1}^{q} A_{ij} Y_j^* = U_i^* , \quad i = 1, \ldots, p \qquad (6.4)$$

from (2.3),

$$\sum_{i=1}^{p} A_{ij} X_i^* = V_j^* , \quad j = 1, \ldots, q \qquad (6.5)$$

from (2.4), and

$$C \cdot X^* = B \cdot Y^* + W^* \tag{6.6}$$

from (2.5) ; also

$$\sum_{k=1}^{m} \Xi_k^* = \sum_{i=1}^{p} X_i^* + \sum_{j=1}^{q} Y_j^* + \sum_{i=1}^{p} U_i^* + \sum_{j=1}^{q} V_j^* + W^* = 1 . \tag{6.7}$$

Now suppose that the pair (X, Y) is a solution of the original LP (2.1)-(2.2) . Filling in with slacks U_i and V_j , we may write

$$\sum_{i=1}^{p} A_{ij} X_i + V_j = B_j , \quad j = 1, \ldots, q , \tag{6.8}$$

$$\sum_{j=1}^{q} A_{ij} Y_j - U_i = C_i , \quad i = 1, \ldots, p , \tag{6.9}$$

$$B \cdot Y = C \cdot X . \tag{6.10}$$

Now multiply the equations (6.8) by Y_j^* and add. Taking account of (6.4), we get

$$B \cdot Y^* = X \cdot U^* + V \cdot Y^* . \tag{6.11}$$

Next, from (6.9) and (6.5), we get

$$C \cdot X^* = - V^* \cdot Y - U \cdot X^* . \tag{6.12}$$

Putting these together with (6.6), we get

$$W^* + X \cdot U^* + U \cdot X^* + Y \cdot V^* + V \cdot Y^* = 0 \quad . \tag{6.13}$$

Since W^* and all the dot products in (6.13) are nonnegative, then they are all zero. Hence, in particular, from (6.11) and (6.12),

$$C \cdot X^* = B \cdot Y^* = 0 \quad . \tag{6.14}$$

Now it is not possible for all the X_i^* and Y_j^* to be zero. If this were so, we would have all the U_i^* equal to zero from (6.4) and all the V_j^* equal to zero from (6.5) . And we have just proved that $W^* = 0$. This would then contradict (6.7).

So suppose that some of the X_i^* are nonzero, and write

$$X^t = X + tX^* \quad , \quad t > 0 \quad . \tag{6.15}$$

Then, by (6.8) and (6.5) ,

$$\sum_{i=1}^{p} A_{ij} X_i^t \leq B_j \quad , \quad j = 1, \ldots, q \quad . \tag{6.16}$$

Hence X^t is primal-feasible. From (6.10) and (6.14),

$$C \cdot X^t = C \cdot X = C \cdot Y \quad . \tag{6.17}$$

It follows that the pairs (X^t, Y) , $t > 0$, form an unbounded solution set for (2.1)-(2.2) . We make the corresponding construction if it is that some of the Y_j^* are nonzero. The lemma is proved. □

The above proof by contradiction does not give any indication of what the bound on that "geometric mean" in (3.26) is. But it does in fact provide, by asserting that some bound exists, a polynomial estimate of the number of steps of the outside algorithm required for given accuracy.

We however do not have a polynomial estimate for the number of arithmetic operations required to achieve given accuracy. This is because the inside algorithm, the steepest-ascent algorithm of [1] whose role is described in §4 above, has an estimate, at (4.24), involving the eigenvalues of a matrix depending on Ξ . We do not even know that the whole series $\{\Xi^N\}$ converges (unless the solution is unique) . So we have no estimate of the smallest eigenvalue α appearing there. All we know is that

that routine yields, in finitely many steps, any desired degree of accuracy. So we will state the overall convergence result, a consequence of the estimate (4.24) in [1] and of the Lemma, very simply as follows.

THEOREM. The algorithm of this paper, which incorporates the steepest-ascent algorithm of [1], will answer to the objective set at (2.7) above in finitely many arithmetic steps. □

§7. The routine

This section is intended to provide a summary guide to the practical execution of the program.

We suppose available the steepest-ascent algorithm of [1], whose action is described there in §§2, 3. In that algorithm there is an objective vector a, defined in some R^K, and constraints b^l, also defined in R^K, with $l = 1, \ldots, L$. The original problem it is concerned with is that of maximizing $a \cdot \gamma$ subject to the constraints $|\gamma| = 1$, $b^l \cdot \gamma = 0$, $l = 1, \ldots, L$. Suppose that maximum is v. The routine finds an approximation to the solution of this problem, in the following sense. Given ϵ positive, it either states that $v \leq \epsilon$, or else produces a unit vector $\gamma' \in R^K$ satisfying $a \cdot \gamma' \geq v$ and $|b^l \cdot \gamma'| < \epsilon$, $l = 1, \ldots, L$.

The main routine is executed relative to strictly positive vectors $\Xi = (X, Y, U, V, W) \in R^m = R^p \times R^q \times R^p \times R^q \times R^1$. It begins at a starting point $\Xi^0 = (X^0, Y^0, U^0, V^0, W^0)$, determined as follows. X^0 and Y^0 are chosen arbitrarily, with all the X^0 and Y^0 strictly positive. For instance one might choose them to be

all equal to unity. The U_i^0 , V_j^0 , and W^0 are then chosen to be strictly positive and such that all the deficits $\Delta_i^0 = \Delta_i(\Xi^0)$, $\Delta_j^0 = \Delta_j(\Xi^0)$, and $\Delta^0 = \Delta(\Xi^0)$ defined at (2.3)-(2.5) are strictly positive and equal. The routine is now ready for the execution of <u>step</u> 0 .

At the outset of step N, $N \geq 0$, the routine is at a point Ξ^N , all of whose components are strictly positive, and for which the deficits $\Delta_i^N = \Delta_i(\Xi^N)$, $\Delta_j^N = \Delta_j(\Xi^N)$, and $\Delta^N = \Delta(\Xi^N)$ are all positive and satisfy (2.6) . The problem is to find a direction for a move on the corresponding simplex Σ^N . We now have to treat the two possibilities considered at the beginning of §4 .

We begin by finding the largest of the deficits Δ_i^N , Δ_j^N , Δ at hand. Suppose its value is $\bar{\Delta}$. We then put

$$U_i^* = U_i + \bar{\Delta} - \Delta_i^N, \quad i = 1,\ldots,p; \quad V_j^* = V_j + \bar{\Delta} - \Delta_j^N, \quad j = 1,\ldots,q; \qquad W^* = W + \bar{\Delta} - \Delta^N . \tag{7.1}$$

At the point $\Xi^* = (X^N, Y^N, U^*, V^*, W^*)$ the deficits are now all equal. Denote its image in Σ^N under T^N by ξ^*. (Recall that T^N is given by (3.1)-(3.2) with Ξ^0 replaced by Ξ^N.) Now test

the inequality (4.1) . If it passes (which seems in general unlikely) , we get an essentially free move. We take γ to be the direction pointing to ξ^* , i.e. $\gamma = (\xi^* - \bar{\xi})/|\xi^* - \bar{\xi}|$, and pass to the description of the "Move on the simplex", below .

If (4.1) fails, we are still guaranteed a balanced reduction, but at substantially greater cost. The algorithm calls the steepest-ascent routine, operating in the space $R^K = R^{2p+2q+2}$ of variables $\xi = (x, y, u, v, w, z)$. The objective vector a is now read off from (3.7), with 0 replaced by N, as follows. We put

$$a_k = C_k X_k^N \ , \ k = 1, \ldots, p : \ a_k = -B_{k-p} Y_{k-p}^N \ , \ k = p+1, \ldots, p+q : \tag{7.2}$$

$$a_k = 0 \ , \ k = p+q+1, \ldots, 2p+2q : \ a_{2p+2q+1} = 0 \ , \ a_{2p+2q+2} = 0 \ .$$

There are $L = p+q+1$ constraint vectors b^ℓ , whose construction requires some intermediate definitions, as follows. We first define vectors $c^i \in R^{2p+2q+2}$, $i = 1, \ldots, p$, read off from (3.5):

$$c_k^i = 0 \ , \ k = 1, \ldots, p : \ c_{k-p}^i = -A_{i, k-p} Y_{k-p}^N \ , \ k = p+1, \ldots, p+q :$$

$$c_k^i = 0 \ , \ k = p+q+1, 2p+2q+1 \ , \ \underline{\text{except that}} \ c_{p+q+i}^i = U_i^0 : \tag{7.3}$$

$$c_{2p+2q+2}^i = C_i \ .$$

Next we define vectors $d^j \in R^{2p+2q+2}$, $j = 1,\ldots,q$, read off from (3.6):

$$d^j_k = A_{kj} X^N_k \ , \ k = 1,\ldots,p \ ; \ d^j_k = 0 \ , \ k = p+1,\ldots,2p+2q+1 \ ,$$

$$\underline{\text{except that}} \ \ d^j_{2p+q+j} = V^0_j \ ; \ d^j_{2p+2q+2} = -B_j \ . \tag{7.4}$$

Next we define scalars

$$\alpha_i = 2\Delta^N_i / \Delta^N - 1 \ , \ i = 1,\ldots,p ; \ \ \alpha'_j = 2\Delta^N_j / \Delta^N - 1 \ , \ j = 1,\ldots,q \ . \tag{7.5}$$

Finally we define

$$b^\ell = c^\ell - \alpha_\ell z \ , \ \ell = 1,\ldots,p ; \qquad b^\ell = d^{\ell-p} - \alpha'_{\ell-p} z \ , \ \ell = p+1,\ldots,p+q . \tag{7.6}$$

These correspond to the constraints (4.6)-(4.7); see the text following those formulas. The final constraint vector has

$$b^{p+q+1}_k = 1 \ , \ \ k = 1,\ldots,2p+2q+1 \ ; \tag{7.7}$$

it is the simplex constraint.

Each iteration of the steepest-ascent algorithm produces a unit vector v', generally speaking not satisfying any of the

constraints. We "plaster" it to the simplex by subtracting, from each component, the average of those components, and then renormalizing. We denote the plastered unit vector by γ. We first test the inequality (4.8), which here reads

$$a \cdot \gamma > |h|/2 \quad , \tag{7.8}$$

where h is the vector, used in the construction of γ', given by (3.1) in [1]. We then make the tests corresponding to (4.9) and (4.10) of this paper. They are:

$$\begin{aligned} b^{\ell} \cdot \gamma &\in [x_{\ell} - 1/4, x^{\ell} + 1/4] , \quad \ell = 1, \ldots, p : \\ b^{\ell} \cdot \gamma &\in [x'_{\ell - p} - 1/4, x'_{\ell - p} + 1/4] , \quad \ell = p+1, \ldots, p+q . \end{aligned} \tag{7.9}$$

If any of these test fails, the steepest-ascent algorithm proceeds to its next iteration. According to the theory (see §4, especially following (4.7), eventually all the tests must pass, and we have the desired direction for a move in the simplex, bringing down $\varphi(\xi) = \delta(\xi)/\rho(\xi)$ and respecting the requirement (3.27) on the condition that the move is of length not over 1/2.

The move on the simplex. Whether we are in case (4.1) or (4.2), we now have a direction γ of guaranteed balanced reduction on the simplex. We seek the approximate minimum for the function $\Phi(t)$ of (5.14). We know from (5.18) that this function is still decreasing when $t = r/4$. So we set the initial left endpoint for the halving process at $a = r/4$. As to a right endpoint, there is first the boundary. Next, in the case (4.1) one cannot move too far beyond ξ^*, precisely not beyond $\xi^{**} = \bar{\xi} + 2(\xi^* - \bar{\xi}) = \bar{\xi} + 2\xi^*$, without running a chance of violating the balance condition (3.27). In the case (4.2) one cannot move more than 1/2 for the same reason. If the effective upper stop is now still the boundary, one uses a halving process, starting with left endpoint at $r/4$ and right endpoint at the boundary, to find a point where $\dot{\Phi}(t) > 0$, and puts b equal to that t. If in case (4.1) it is ξ^{**} that is at the stop, we test the sign of $\dot{\Phi}(t)$. If that is negative, we put $\xi^{\#} = \xi^{**}$ and pass to the projective transformation. If it is negative, we put $b = \xi^{**}$ and pass to the halving process. If in case (4.2) the upper stop is $d/2$, we proceed similarly.

At this point, if we have not found $\xi^{\#}$ from either of the two special cases noted above, we have a lower limit $a = r/4$, and an upper limit b, for the minimum. We then begin the

halving process, testing the midpoint and at each step redefining a or b . If in the unlikely event that we find a point where $\dot{\Phi}(t) = 0$, the process stops . Supposing it does not stop that way, we run it a few iterations, perhaps ten or twenty ; there is no reason to seek high precision here. Then either the ξ at the stop, or the ξ corresponding to $d = (a+b)/2$ will be taken to be $\xi^{\#}$.

The projective transformation . This is very easily executed. One simply uses (3.11), with N replaced by $N+1$, to define Ξ^{N+1} from Ξ^{N} and $\xi^{\#}$. One is then ready to execute step $N+1$. The algorithm terminates when the criterion (2.7) is met. □

LITERATURE [References incomplete]

1. John M. Danskin, A geometric algorithm for steepest ascent with linear constraints, Unpublished M/S , June 1 1985

2. Narendra Karmarkar, the first unpublished M/S circulared from AT&T , 1984

3. Narendra Karmarkar, the published M/S , Combinatorica , 1985

ENCLOSURE C

AN EXPERIMENTAL APPROACH TO KARMARKAR'S PROJECTIVE METHOD FOR LINEAR PROGRAMMING

BY J.A. TOMLIN

An Experimental Approach to Karmarkar's Projective Method for Linear Programming*

by

J.A. Tomlin

Ketron, Inc.

Mountain View, CA 94040

Abstract

This paper describes the evolution of an experimental implementation of the Karmarkar projective method and gives computational results for some small to medium, but realistically structured models.

Key words: Linear Programming, Projective Method, Simplex Method, Linear Least Squares, Sparse Matrices

January, 1985

*This paper supersedes an earlier version entitled "An Experimental Approach to Karmarkar's Linear Programming Algorithm."

An Experimental Approach to Karmarkar's Projective Method for Linear Programming

by

J.A. Tomlin

Introduction

A September 1984 article in Science [13] made breathless claims for the speed and efficiency of a new linear programming (LP) algorithm devised by N. Karmarkar [10]. Furthermore, claims were made that numerical results prove the new method up to 50 times faster than the simplex method. Unfortunately, no information on the test problem(s) or experimental procedures were given. Only limited details of the new implementation have so far been discussed publicly.

At first sight, the prospects for such a new algorithm are not bright. Thirty-five years of attempts to defeat the simplex method [2] have met with uniform lack of success, the most recent being the Scolnik and ellipsoid fiascos. This generally gloomy outlook is complicated by two further factors implicit in Karmarkar's approach:

(1) Like the ellipsoid method, it takes the unpromising step of non-linearizing a linear problem.

(2) At least initially, it makes the entirely false assumption that any feasible LP constraint set possesses a strict interior (i.e., a solution positive in every component). Many real LPs are highly degenerate [17].

It seems that the only way to evaluate these conflicting viewpoints is to perform some computational experiments on reasonably representative LP

models and extrapolate the results. This paper reports on the results of such a set of experiments. The next section reviews the steps of the algorithm as outlined in [10]. The following sections describe some additional constructive steps for getting started, the implementation, experimental results and conclusions.

1. The Projective Algorithm

While some familiarity with Karmarkar's paper is important, we include the following brief outline of the projective algorithm as presented in [10]:

The LP is assumed to be cast in the almost homogeneous form:

$$\text{Min } z = c^T x \tag{1a}$$

subject to

$$Ax = 0 \tag{1b}$$

$$e^T x = 1 \quad , \qquad x \geq 0 \tag{1c}$$

where e is an n-vector of 1's, A is $m \times n$, and the minimum z is assumed to be 0. It is also assumed that there exists a solution to (1) with the property that $\bar{x}_j > 0$ $(j = 1, \ldots, n)$.

Under these assumptions, let $D = \text{diag}(\bar{x}_1, \ldots, \bar{x}_n)$ and employ a projective transformation and its inverse, defined by:

$$x' = \frac{D^{-1}x}{e^T D^{-1} x} \quad , \qquad x = \frac{Dx'}{e^T Dx'} \quad . \tag{2}$$

The second of these transforms the LP on a simplex (1) into the fractional program on a simplex in x'-space:

$$\text{Min } z = \frac{c^T Dx'}{e^T Dx'} \tag{3a}$$

subject to

$$ADx' = 0 \tag{3b}$$

and

$$e^T x' = 1 \quad , \qquad x' \geq 0 \quad . \tag{3c}$$

Note that from the definition of D, the point $\bar{x}$ in x-space is mapped onto the point $(1/n, \ldots, 1/n)$ in x'-space. The general idea is now to ignore the denominator in (3a), and take a "large" improving step away from the center of the simplex to a new point in x'-space. This is transformed back into x-space and the result evaluated.

Specifically, the algorithm generates a sequence of points $x^{(1)}$, $x^{(2)}$, ..., $x^{(k)}$, where $x_j^{(1)} > 0$ $(j = 1, \ldots, n)$ as follows:

1. Define $D = \text{diag}\,(x_1^{(k)}, \ldots, x_n^{(k)})$, and:

$$B = \begin{bmatrix} AD \\ \hline e^T \end{bmatrix}$$

2. Compute:

$$c_p = [I - B^T(BB^T)^{-1}B]\, Dc \ , \tag{4}$$

(i.e., project an "ascent" direction for problem (3) into the null-space of its constraint matrix B).

3. Normalize c_p and scale it by the radius of the largest sphere which can be inscribed in the simplex (3c) to produce the direction vector:

$$p = \frac{c_p}{|c_p|\sqrt{n(n-1)}} \ . \tag{5}$$

4. Take a "descent" step of length α to a new feasible point for (3):

$$x' = \frac{1}{n} e - \alpha p \ . \tag{6}$$

5. Project x' back into x-space to obtain the new point

$$x^{(k+1)} = \frac{Dx'}{e^T Dx'} \ . \tag{7}$$

If the new point satisfies the termination criterion, stop. Otherwise, set $k \leftarrow k+1$ and go to 1.

It should be clear that the great majority of the work in each iteration comes in step 2, which ensures feasibility of the new point.

2. Getting Started

There are two details to be cleared up in obtaining a first feasible interior-point solution to LP (1). First of all, the problem must be "homogenized." LP problems are more commonly expressed in a canonical form such as:

$$\text{Min } z = c^T x$$

subject to

$$Ax = b \tag{8}$$

$$x \geq 0 .$$

Karmarkar suggested scaling the variables by some plausible upper bound (say σ) on their sum, so that:

$$A\bar{x} = \bar{b} = b/\sigma \tag{9}$$

$$e^T\bar{x} + \bar{x}_{n+1} = 1$$

and then multiplying $\bar{b}$ by $e^T\bar{x} = 1$ to obtain

$$([A,0] - \bar{b}e^T)\,\bar{x} = 0 \tag{10}$$

$$e^T\bar{x} = 1$$

where e and $\bar{x}$ here are of dimension $n+1$. Since b is normally much denser than the columns of A explicit construction of (10) is to be avoided if possible.

One acceptable method is to define a new variable corresponding to the right hand side in (9) after scaling, and forcing this variable to be 1:

$$\begin{aligned} A\bar{x} - (b/\sigma)\xi &= 0 \ , \\ \xi &= 1 \ , \\ e^T\bar{x} + \bar{x}_{n+1} &= 1 \ . \end{aligned} \tag{11}$$

The last two constraints may be replaced by:

$$e^T\bar{x} - \xi + \bar{x}_{n+1} = 0$$

$$e^T\bar{x} + \xi + \bar{x}_{n+1} = 2 \ .$$

Scaling all the variables again by $1/2$ we obtain:

$$\begin{bmatrix} A & -b/\sigma & 0 \\ e^T & -1 & 1 \\ e^T & 1 & 1 \end{bmatrix} \hat{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \tag{12}$$

$$\hat{x} \geq 0$$

and a solution to (8) may be recovered as $x_j = 2\sigma \hat{x}_j$ $(j = 1, \ldots, n)$. Note that A is unaffected, but a single dense row is added.

Karmarkar also suggested the construction of an artificial column to obtain a starting interior solution to (1). This involves adding a new column (say d) and variable to (1b) and (1c) —

$$Ax + d\lambda = 0$$

$$e^T x + \lambda = 1 \quad ,$$

such that $e/(n+1)$ is a feasible solution, and then minimize λ to zero (or close to zero). Using the construction (12):

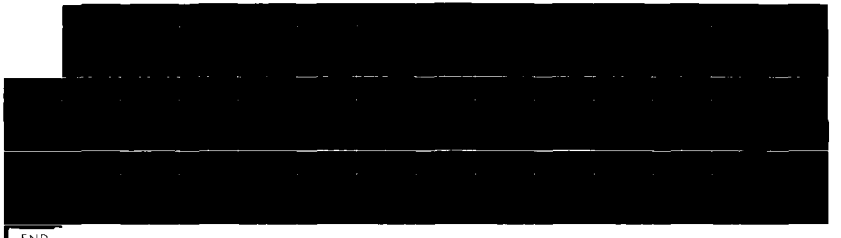
$$\begin{bmatrix} A & -b/\sigma & 0 & d \\ e^T & -1 & 1 & \delta \\ e^T & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \qquad (13)$$

and retaining the convention that A is $m \times n$, a starting solution of $e/(n+3)$ implies that $d = b/\sigma - Ae$, $\delta = -n$.

For these experiments we thought it better to start with a different interior point. We already know that the variable $\hat{x}_{n+1}$ corresponding to the right hand side must be $1/2$ in a feasible solution — quite distant from $1/(n+3)$. In addition, the slack value $\hat{x}_{n+2}$ may be substantial,

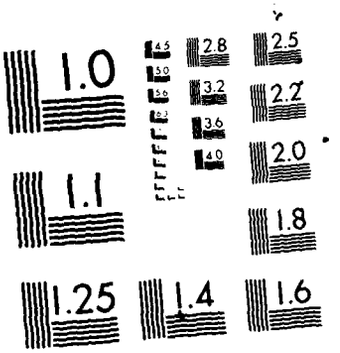
AD-A172 679 AN INVESTIGATION OF KARMARKAR'S ALGORITHM FOR LINEAR PROGRAMMING(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA G E PUGH ET AL. 10 SEP 86 DSA-707 2/2

UNCLASSIFIED N00014-85-C-0254 F/G 12/1 NL



END

DATE FILMED 12-86



since the value σ will rarely be known accurately. Finally, an initial value of $1/(n+3)$ for λ gives the algorithm very little to "bite" on, since this is not very far from the target value of zero (or ϵ) for even moderate n.

We have used the starting solution:

$$\hat{x}_j = \frac{1}{4n} , \qquad (j = 1, \ldots, n)$$

$$\hat{x}_{n+1} = \hat{x}_{n+2} = \lambda = \frac{1}{4} , \tag{14}$$

which gives the artificial coefficient values:

$$d = \frac{b}{\sigma} - \frac{Ae}{n} , \qquad \delta = -1 .$$

This has proved satisfactory in practice.

3. Implementation

To initiate these experiments the author's LPM1 Fortran LP code was modified to give a test bed implementation (LPMK) of Karmarkar's algorithm for sparse LP problems of moderate size. (We had previously modified this code to test Scolnik's approach [15]).

The matrix $[A, -b/\sigma]$ has its non-zeros stored contiguously column-wise in arrays $LA(\cdot)$, $A(\cdot)$, specifying the row indices and values. Another array $LA(\cdot)$ points to the beginning of each column. Slack columns are stored for each L and G row in the conventional MPS format [12], but no logical column is generated for an E (equality) row. All

free rows are dropped, except the designated cost row, which is stored as a full vector. All of these arrays are constructed as the LP model is read in MPS format. Note, however, that bounds and ranges are not allowed.

The artificial column is constructed immediately after input, but the two dense rows $(m+1)$ and $(m+2)$ are never stored explicitly.

As we have pointed out, the bulk of the work in the algorithm is the calculation of a projected vector (4). It turns out that the calculation (4) is equivalent to finding the residual of a least squares problem:

$$\bar{y} = \arg\min_{y} \|Dc - B^T y\|_2 \tag{15a}$$

$$c_p = Dc - B^T\bar{y} \ . \tag{15b}$$

There are several methods of computing this vector (see [6,7,8]). The method chosen for this implementation is the popular one of computing a matrix decomposition:

$$QB^T = \begin{bmatrix} R \\ 0 \end{bmatrix} \ , \ QDc = \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \end{bmatrix} \tag{16}$$

where R is upper triangular, $Q^TQ = I$ and B is initially assumed of full rank. c_p may then be computed via $\bar{y} = R^{-1}\hat{c}_1$ and (15b) or:

$$c_p = \left[I - B^T R^{-1} R^{-T} B\right] Dc \ . \tag{17}$$

(Alternative procedures will be discussed below.)

In our initial code, Q was computed as a product of Householder transformations:

$$Q = H_t \cdots H_2 H_1 ,$$

where

$$H_k = I - \beta_k w^{(k)} w^{(k)T}$$

and

$$H_k^T H_k = I .$$

Each transformation H_k eliminates the elements (k+1), (k+2), ... of column k of $H_{k-1} \cdots H_2 H_1 B^T$.

Some attempt to exploit structure was made by permuting B^T in the general form of Figure 1 (see Saunders [16]). Unfortunately, b is usually quite dense, and d often completely dense, and any QR decomposition of B^T results in a completely dense R as well as dense $w^{(k)}$ vectors to be stored for the H_k.

The calculations involved in computing the descent direction in (17) are simply routine sparse matrix multiplication and forward and back substitution with R.

4. Experimental Procedures

Table 1 displays the characteristics of 11 test problems. The first two are merely small text book models. GNET20 is a 20 node generalized network (transportation) problem. STD23 is the "standard 23 row refinery model" used for many years as an illustrative model. AFIRO, ADLITTLE, SHARE2B, ISRAEL, BRANDY, E226, and BANDM are realistic models which have been used quite extensively as test problems. Note that the non-zero count includes the slack columns and right hand side, but not the two dense rows and the artificial column d in (13).

Table 2 gives solution data for the models using Katron's MPSIII system [12]. The times include initialization, CONVERT, SETUP, solving via WHIZARD and SOLUTION. The time quoted is for the entire job step on an IBM 3033/N. The number of "eta non-zeros" given refers to the number produced by the final INVERT before the solution is printed. The "Crashed" columns are made basic by inspection in WHIZARD before beginning the simplex iterations.

Two important parameters are needed for the projective algorithm:

(a) A convergence tolerance must be specified. The objective functions of the test problems have been translated by their known optima so that they assume their minima at zero (see [10]). All the problems except ADLITTLE and ISRAEL have optimal values of order close to 10^3. Those problems had their objectives scaled by 10^{-3} to bring them into this range, and the convergence criterion for the translated optima was arbitrarily set to 10^{-6} for <u>all</u> runs.

(b) The length of the step α must be specified or computed. Karmarkar's paper [10] implies that α should be some constant less than 1 (in fact 0.25 is suggested as "safe"). Experiments using fixed values of α for two problems showed the number of iterations required for convergence to be inversely proportional to α (see Table 3). An initial (constant) setting of $\alpha = 0.9$ seemed "safe," and was adopted.

Table 4 gives solution data for this version of the projective method. Two factors other than iteration numbers and times are of particular interest. These are the number of non-zeros in the Q and R factors and the approximate condition of BB^T, which is critical for the accuracy

of the projection step (4). The condition of R is approximated by taking the ratio of the maximum diagonal element $|r_{ii}|$ and the minimum $|r_{ii}|$. This ratio is squared to approximate $\kappa(BB^T)$. In general the number of non-zeros in Q and R decreases throughout the run, while the condition number increases monotonically.

These initial results show three important trends:

(a) A not-unexpected deterioration in the condition of R (and hence BB^T) as the algorithm progresses.

(b) A disappointingly large, but quite slowly growing, number of iterations with problem size.

(c) Catastrophic Fill-In of Q and R.

Ill-conditioning of R is to be expected as diagonal elements of D approach zero and hence B approaches column rank difficiency (especially for degenerate models). A may also be row rank deficient initially, since we do not necessarily have a full set of unit columns. We must therefore have a method of rejecting rows of A (columns of B^T) for elimination in (16). This may, of course, be done by means of a tolerance on acceptable values of $|r_{ii}|$. A value of 10^{-12} is used here for all elements of Q and R. Some more sophisticated method is clearly desirable.

The ill-conditioning of R can sometimes be overcome simply by avoiding its use. M.A. Saunders suggested using the formula:

$$c_p = Q^T \begin{bmatrix} 0 & \\ & I \end{bmatrix} QDc \ , \qquad (18)$$

(where the 0 diagonal matrix is of the row dimension of B), rather than (15b) or (17), involving only the necessarily well-conditioned Q. Experiment showed that this did indeed allow convergence to tighter tolerances, though (18) may require prohibitively more work if the number of non-zeros in Q is much greater than in R and B.

5. Modified Step Length Calculation

The number of iterations, as we have observed, tends to be inversely proportional to α. There is in fact no good reason to restrict α to be less than 1, except for proving theoretical complexity results [10]. What we do require, from (6), is that:

$$\alpha \leq \phi = \min_{p_i > 0} \frac{1}{np_i} \quad . \tag{19}$$

We would also like to guarantee that $z^{(k+1)} < z^{(k)}$. This may not happen in some circumstances. In particular it was observed that when some of the problems were infeasible (due to input errors) and α was fixed ($0 < \alpha < 1$) the phase I objective function λ oscillated, and these oscillations did not converge. This un-robust behavior suggest a look at the conditions under which we expect $z^{(k)}$ to improve.

Defining

$$\mu = n \sum_j c_j x_j^{(k)} p_j \quad , \quad \nu = n \sum_j x_j^{(k)} p_j \quad , \tag{20}$$

we easily see from (1a), (6) and (7) that

$$x_j^{(k+1)} = x_j^{(k)} \left\{ \frac{1 - n\alpha\, p_j}{1 - \alpha\, \nu} \right\} , \tag{21}$$

$$z^{(k+1)} = \frac{z^{(k)} - \alpha\, \mu}{1 - \alpha\, \nu} . \tag{22}$$

Clearly $z^{(k+1)}$ is monotonic in α, but a decrease is not guaranteed. The four cases which do satisfy $z^{(k+1)} < z^{(k)}$ for some range of α are:

$\mu - \nu\, z^{(k)} > 0$ and:

(i) $\nu > 0 \Rightarrow 0 < \alpha < 1/\nu$,

(ii) $\nu < 0 \Rightarrow 0 < \alpha$,

$\mu - \nu\, z^{(k)} < 0$ and:

(iii) $\nu > 0 \Rightarrow \alpha < 0$,

(iv) $\nu < 0 \Rightarrow 1/\nu < \alpha < 0$.

The oscillations must be due to cases (iii) or (iv) arising when we insist that α be a positive constant.

The oscillating case may be avoided by setting $\alpha < 0$ is cases (iii) and (iv). Note that this has <u>never</u> been observed to happen in practice unless a model is infeasible or the constant σ in (9) is chosen to be too small.

The α calculation to be used in practice is then to specify a multiplier $0 < \bar{\alpha} < 1$ and in cases (i) and (ii) above set:

$$\text{(i)}\ \alpha = \bar{\alpha}\ \min\{\phi,\ 1/v\} \qquad (v > 0)$$

$$\text{(ii)}\ \alpha = \bar{\alpha}\ \phi \qquad (v < 0) \tag{23}$$

otherwise we simply put $\alpha = -\bar{\alpha}$ in case (iii) or $\alpha = \min\{-\bar{\alpha},\ 1/v\}$ in case (iv).

The problem of fill-in was left temporarily in abeyance and the set of problems in Table 4 rerun with the revised step length calculation (23).

The number of iterations in Table 5 is reduced by a factor of about 4 for the larger models. There does seem to be some loss of accuracy in that there was difficulty in attaining convergence for problem SHARE2B. This was immediately overcome by use of formula (18).

6. Reducing Fill-in

The presence of the right hand side column b in B, and of d in phase I, almost guarantees complete fill-in of R (and of Q within the envelope implied by an ordering such as seen in Figure 1). This catastrophic fill-in can be avoided by initially omitting the offending columns from B (i.e., rows from B^T) and using an "update" to compute the least squares solution.

If $B = [B_1\ B_2]$, where B_2 contains the "nasty" columns, the procedure is to form the modified factorization

$$\hat{Q}\ B_1^T = \hat{R}\ ,$$

and solve a modified least squares problem to obtain:

$$\hat{y} = \underset{y}{\operatorname{argmin}} \| D_1 c_1 - B_1^T y \|_2 \quad ,$$

where D and c are partitioned conformably with B.

There are a number of ways of modifying this solution $\hat{y}$ to obtain the solution to the full problem (15) (see e.g., [3]). In its simplest form this updating requires us to calculate:

$$\begin{aligned} F &= B_2^T \hat{R}^{-1} \quad , \qquad f = D_2 c_2 - B_2^T \hat{y} \\ \bar{y} &= \hat{y} + \hat{R}^{-1} F^T v \quad , \end{aligned} \tag{24}$$

where v solves:

$$(I + FF^T)v = f \quad . \tag{25}$$

Ignoring the work required in solving (25), this calculation is dominated by a forward substitution using $\hat{R}$ for each row of B_2^T and a back substitution for $F^T v$. The assumption is that the total work will be less than that using QR when R becomes dense. Note that only the updated solution $\bar{y}$ for (15a) is obtained, not an updated factorization.

7. Modified Phase I

In addition to the above feature, another improvement was made, which can reduce the number of iterations in phase I. Karmarkar has pointed out [11] that λ need not be non-negative and can be omitted from the step length calculation (19). If the α chosen results in $\lambda < 0$, one can interpolate between $x^{(k)}$ and $x^{(k+1)}$ to find an x such that

$\lambda = 0$. In our terms, using λ for $x^{(k)}$ in (21), and letting p_d correspond to the component of p for the artificial column d, if

$$\frac{1 - n\,\alpha p_d}{1 - \alpha v} < 0 ,$$

then reset α to $1/(np_d)$.

8. Minimum Degree Ordering

Once care is taken to avoid fill-in in R (actually $\hat{R}$) it makes sense to use a more sophisticated ordering than that employed so far. The method chosen was the "minimum degree" ordering of AA^T (see [4]), since the non-zero structure of L^T, where $P^TAA^TP = LL^T$, is the same as the non-zero structure of R when we compute $QDA^TP = \begin{bmatrix} R \\ 0 \end{bmatrix}$.

This ordering requires finding the non-zero structure (not the values) of AA^T, that is its "adjacency matrix," and then application of the minimum degree algorithm to this structure. We used the data structures and procedures in [4] without modification and attained very satisfactory results. The only exception to use of this ordering was for problem SHARE2B, which turns out to have a natural "block angular" ordering with dense diagonal blocks above the linking rows. This structure is almost ideal, and is used as it stands.

The non-trivial models in Table 5 were run again with the three enhancements in Sections 6-8, and the results are displayed in Table 6. The improvement is substantial, and the times for this subset of models begins to bear comparison with the MPSIII times.

9. Use of Givens Rotations

Attempting to build on this success and solve model E226 met immediate difficulty. For larger problems, trying to perform the QR decomposition using Householder transformations becomes impractical as the number of intermediate and final non-zeros becomes very large, as illustrated by Heath [8]. The method of George and Heath [3] overcomes this by never storing Q and using Givens rotations to eliminate rows of B^T, building R with a row-wise data structure.

Using the George and Heath technique we are able to make use of the full power of the minimum degree ordering and symbolic factorization procedures described by George and Liu [4]. The adjacency matrix is constructed as before, and the minimum degree ordering (of the columns of A^T) is found. This is followed by a symbolic factorization which predicts where all the non-zeros in R will be (assuming no cancellation). The data structure for this (mildly pessimistic) estimate of the sparsity pattern of R is set up once-and-for-all, and used in all subsequent factorizations (at least in this implementation). Note that since Q is discarded, we update the Dc vector simultaneously with the factorization, computing:

$$Q\left[DA^TP \;\vdots\; De \;\vdots\; e \;\vdots\; Dc\right] = \begin{bmatrix} R & \vdots & \\ 0 & \vdots & QDc \end{bmatrix} . \qquad (26)$$

Using this approach, we were able to run all the models in Table 1, with results as shown in Table 7 (omitting the first two models).

There are some differences in the data displayed. Since we work only with R, obtaining c_p directly as in (15b), the estimate of $\kappa(R)$, not

its square, is given. There is now only one number for R non-zeros, as computed in the symbolic factorization of AA^T. (This does not include diagonal non-zeros of R or those in the last two columns.) The size of the step length α actually attained is also of interest (note that $\bar{\alpha} = 0.99$ is used here).

10. Discussion of Computational Results

It is clear that the Givens-George-Heath approach has again resulted in substantial improvement, and enabled the solution of the larger models in tolerable — if not competitive — time.

The very bad performance on problem ISRAEL is due to the presence of three very dense columns (as well as several dense rows) in the LP matrix. This difficulty could be partly overcome by using an update to introduce these columns — that is including them as "nasty" columns along with b and d. The present implementation does not allow this. However if these dense columns are omitted from the matrix the number of non-zeros in (symbolic) R drops to 4525, indicating that a two- or three-fold improvement may be possible. However, there must clearly be a point at which the extra work in the updating approach begins to outweigh the savings in sparsity of R. Note that the simplex method had no particular difficulty with this problem.

The number of iterations appears to grow quite slowly with model size. Karmarkar [11] has claimed that the number of iterations should be O(log n). While we would not make so formal a claim, these results by no means contradict it.

An encouraging sign is the relative sparseness of R (except for problem ISRAEL). The number of non-zeros is generally only 2 or 3 times the number of non-zeros in the factors of the optimal basis. Given the vulnerability of QR decomposition to fill-in, this speaks well of the minimum degree ordering. Since the method behaved well in the case of block angular A (as should be expected) it suggests that nested dissection [4], or incomplete nested dissection might also be worth investigating.

The relative sparsity of the resulting R must not, however, be confused with the amount of work required to obtain it via QR decomposition of B^T. Theoretical considerations, and our numerical results, indicate that this can grow drastically with n, even using a close to "state of the art" direct method (see [5] for later developments).

11. Finding "Exact" Optimum Data

In practice, one must ask how a usable optimum basic feasible solution to the LP (and its dual) should be extracted when the projective method is deemed to have converged. One might hope to use the projective method to "front-end" the simplex method. As it happens, many MP systems contain a procedure quite suitable in principle for this task. In all descendents of MPS/360 (including MPSIII and MPSX/370) this is known as the BASIC procedure. Benichou et al. give a general description of this method in section 2.7 of [1] (see also [9,12]). Essentially, BASIC accepts a non-basic solution to a LP and transforms it into a basic solution whose objective (phase I or II) value is at least as good. This may be followed by the simplex method to attain optimality.

Experiments with the first (fixed α, Householder) version of the code indicated that the projective method could provide a reasonably good starting solution, via BASIC, even with loose phase II convergence criteria. Thus Table 8 gives data on the BASIC and simplex steps required to optimize SHARE2B when the projective method is stopped early and at termination. It is noticeable that feasibility in the projective method deteriorates toward the optimum, and early termination provides a satisfactory starting solution for BASIC. For larger problems, and with a longer step length (fewer iterations) this deterioration is more marked, and it proved difficult to get good starting solutions for BASIC. Indeed BASIC and subsequent simplex iterations were sometimes at many as those required to solve the problem without the projective method as a "front-end." This suggests that an additional iteration (or more) should be carried out at "convergence" to enforce feasibility, or that perhaps some "composite" steps should be performed, which attempt to restore feasibility as the optimum is approached. This has not been attempted in this series of experiments.

12. Conclusion

Following an experimental trail beginning with [10], we have been able to implement a version of the projective method which solves realistic LP models in respectable, in not really competitive, times. This is not to say that further improvements are not possible. Karmarkar has hinted [11] that Bell Labs are experimenting with the use of incomplete Choleski factorization of BB^T to precondition a conjugate gradient method [7] for solving the least squares problems at the core of this method. This promises to lead to some improvement over the direct QR methods to which

these experiments have been confined. An even more promising approach is some preconditioned form of the LSQR algorithm [14].

Even though more sophisticated schemes for solving sparse least squares problems may lead to markedly improved performance, it is difficult to see how they can lead to 50 to 1 improvements over the simplex method except in very special cases. As it happens, the model for which this claim was made is reported to be a multicommodity flow problem [11], which is block-angular and has an otherwise remarkably "friendly" matrix.

Experience shows that it is not difficult to beat MPSX/370 (the chosen yardstick) by a wide margin on embedded network models (of which the multicommodity problem is a special case). We have reported results [18] for a model with about 5000 network rows and 700 "hard" rows where exploitation of the network simplex method in MPSIII enabled us to solve it in about 5 minutes of IBM 3033 time, when MPSX/370 — used naively — had failed to terminate in over an hour. Other practitioners were able to tie a different network optimizer in with MPSX/370 and solve the model in about 11 minutes.

Finally, our observations may be summarized as follows:

(1) The number of iterations required by the projective method does indeed grow slowly with model size.

(2) The work per iteration is substantial, and limited by the technology for solving sparse least squares problems [8].

(3) Vast improvements in speed over the simplex method only seem possible for special classes of problems, and these may be problems for which greatly improved simplex technology is already available.

Acknowledgement

The author is greatly indebted to M.A. Saunders for encouragement, suggestions, and many helpful discussions in the course of this research.

References

[1] M. Benichou, L.M. Gauthier, G. Hentges and G. Ribiere, "The Efficient Solution of Large-Scale Linear Programming Problems — Some Algorithmic Techniques and Computational Results," Math. Prog. 13, 280-322 (1977).

[2] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, N.J. (1963).

[3] A. George and M.T. Heath, "Solution of Sparse Linear Least Squares Problems Using Givens Rotations," Linear Algebra Appl., 34, 69-83 (1980).

[4] A. George and J. W.-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, N.J. (1981).

[5] A. George and E. Ng, "A New Release of SPARSPAK — The Waterloo Sparse Matrix Package," ACM SIGNUM Newsletter, Vol. 19, No. 4, 9-13, October 1984.

[6] P.E. Gill, W. Murray and M.H. Wright, Practical Optimization, Academic Press, London (1981).

[7] G.H. Golub and C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, Md. (1983).

[8] M. Heath, "Numerical Methods for Large Sparse Least Squares Problems," SIAM J. of Sci. & Stat. Comp. 5, 497-513, (1984).

[9] IBM Corporation, "Mathematical Programming System/360, Version 2, System Manual," Form Y20-0065-2, White Plains, N.Y. (1969).

[10] N. Karmarkar, "A New Polynomial Time Algorithm for Linear Programming," AT&T Bell Labs, Murray Hill, N.J., Undated, c. 1984.

[11] N. Karmarkar, Seminar presentations at ORSA/TIMS, Dallas, TX, November 1984, and Stanford, CA, January 1985.

[12] Ketron, Inc., "MPSIII Users Manual," Revision 11, Arlington, Va., June, 1984.

[13] G. Kolata, "A Fast Way to Solve Hard Problems," Science, 225, 1379-1380, September 21, 1984.

[14] C.C. Paige and M.A. Saunders, "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," ACM Trans. on Math. Softw., 8, 43-71 (1982).

[15] M. Saiidi and J.A. Tomlin, "Some Computational Experiments with Scolnik's Linear Programming Approach," SIGMAP Newsletter, 18, 30-37 (1975).

[16] M.A. Saunders, "Large Scale Linear Programming Using the Cholesky Factorization," STAN-CS-72-252, Computer Science Department, Stanford University, January, 1972.

[17] J.A. Tomlin and J.S. Welch, "Formal Optimization of Some Reduced Linear Programming Problems," Math. Prog. 27, 232-240 (1983).

[18] J.A. Tomlin and J.S. Welch, "Integration of a Primal Simplex Network Algorithm with a Large Scale Mathematical Programming System," to appear in ACM Trans. on Math. Softw.

Name	Constraints	Slacks	Columns	Non-Zeros	Optimum	σ
HEXNUT	3	3	4	15	11.0	16.0
FEEDMIX	4	3	3	16	5.2857	16.0
GNET20	20	0	44	108	760.179	16^2
STD23	22	12	23	103	-45.53024	16^3
AFIRO	27	19	32	110	-494.75293	16^3
ADLITTLE	56	41	97	461	225494.94	16^3
SHARE2B	96	83	79	801	-415.73224	16^3
ISRAEL	174	174	142	2274	896644.8	16^4
BRANDY	220	54	249	2256	1518.51	16^4
E226	223	190	282	2876	-18.7519	16^3
BANDM	305	0	472	2612	158.628	16^4

Table 1. Test Problem Statistics

Name	"Crashed" Cols.	Simplex Its.	Eta Non-Zeros	Time (Secs.)
HEXNUT	2	1	6	0.58
FEEDMIX	1	3	14	0.59
GNET20	19	20	69	0.72
STD23	10	6	72	0.52
AFIRO	18	4	52	0.53
ADLITTLE	18	80	266	1.02
SHARE2B	13	84	558	1.06
ISRAEL	1	166	1480	2.35
BRANDY	85	168	1467	2.66
E226	50	350	1490	3.95
BANDM	114	253	2385	3.51

Table 2. MPSIII/WHIZARD Solution Results

	Iterations to Optimality	
α	STD23	AFIRO
0.25	165	239
0.5	82	118
0.9	44	64

Table 3. Sensitivity to (Fixed) α

Name	Iterations to:		Q Non-Zeros		R Non-Zeros		Appr. Cond. BB^T		Time (Secs.)
	Feasibility	Optimality	Min	Max	Min	Max	Min	Max	
HEXNUT	7	25	32	37	12	25	10^3	10^3	0.18
FEEDMIX	8	20	29	35	21	21	10^3	10^5	0.17
CNET20	46	58	391	416	253	253	10^6	10^{16}	1.30
STD23	17	44	292	351	243	278	10^6	10^{17}	1.01
AFIRO	15	64	358	489	225	408	10^6	10^{16}	1.65
ADLITTLE	29	112	2969	3060	1515	1711	10^7	10^{22}	30.93
SHARE2B	32	89	4496	4862	3990	4851	10^{10}	10^{14}	67.81

Table 4. Initial Projective Algorithm Results

Name	Iterations to:		Q Non-Zeros		R Non-Zeros		Appr. Cond. BB^T		Time (Secs.)
	Feasibility	Optimality	Min	Max	Min	Max	Min	Max	
HEXNUT	6	16	32	37	12	25	10^3	10^3	0.16
FEEDMIX	6	15	29	35	21	21	10^3	10^5	0.16
GNET20	11	19	391	416	253	253	10^6	10^{16}	0.55
STD23	7	16	290	351	226	278	10^6	10^{17}	0.48
AFIRO	7	19	356	489	225	408	10^6	10^{17}	0.62
ADLITTLE	7	26	2919	3060	1515	1711	10^7	10^{22}	7.31
SHARE2B	9	42[a]	3949	4862	3926	4851	10^{10}	10^{20}	29.38
"	9	22[b]	4066	4862	-	-	10^{10}	10^{18}	16.91

[a]Using formula (17). Iterations after about number 22 should be discarded.

[b]Using formula (18).

Table 5. Revised Step Length Results ($\bar{\alpha} = 0.9$)

Name	Iterations to:		Q Non-Zeros		R Non-Zeros		Appr. Cond. BB^T		Time (Secs.)
	Feasibility	Optimality	Min	Max	Min	Max	Min	Max	
GNET20	11	23	198	223	92	117	10^3	10^{15}	0.54
STD23	4	13	194	213	136	140	10^5	10^{17}	0.36
AFIRO	3	15	296	321	152	164	10^5	10^{16}	0.48
ADLITTLE	7	24	1483	1589	483	526	10^7	10^{20}	2.88
SHARE2B	7	19	1984	2594	1100	1216	10^7	10^{13}	5.20

Table 6. Enhanced Householder Version ($\bar{\alpha} = 0.9$)

Name	Iterations to:		R Non-Zeros	Approx. Cond. R		α		Time (Secs.)
	Feasibility	Optimality		Min	Max	Min	Max	
GNET20	10	15	54	10^1	10^8	1.08	4.28	0.35
STD23	4	11	71	10^2	10^9	1.58	3.77	0.31
AFIRO	3	14	80	10^3	10^8	1.26	4.02	0.40
ADLITTLE	5	19	355	10^3	10^{11}	1.41	8.91	1.87
SHARE2B	6	17	1038	10^5	10^7	1.80	7.87	2.80
ISRAEL	9	21	11259	10^4	10^7	1.23	10.6	55.07
BRANDY	11	21	3251	10^5	10^{10}	1.67	9.94	17.23
E226	10	27	3416	10^5	10^{11}	3.08	12.8	31.66
BANDM	12	35	4355	10^5	10^{10}	1.35	11.0	33.13

Table 7. Final Givens Version ($\bar{\alpha} = 0.99$)

Projective Iterations	Time (Secs.)	At Entry to BASIC			After BASIC	BASIC Iters.	Additional Simplex Its.	Time (Secs.)
		Sum Infeas.	# Infeas.	Objective	Objective			
50	39.27	2.1051	16	-408.622	-415.348	74	2	0.73
89	67.81	2.8762	28	-415.735	-407.322	59	2	0.71

Table 8. "BASIC" Runs for SHARE2B

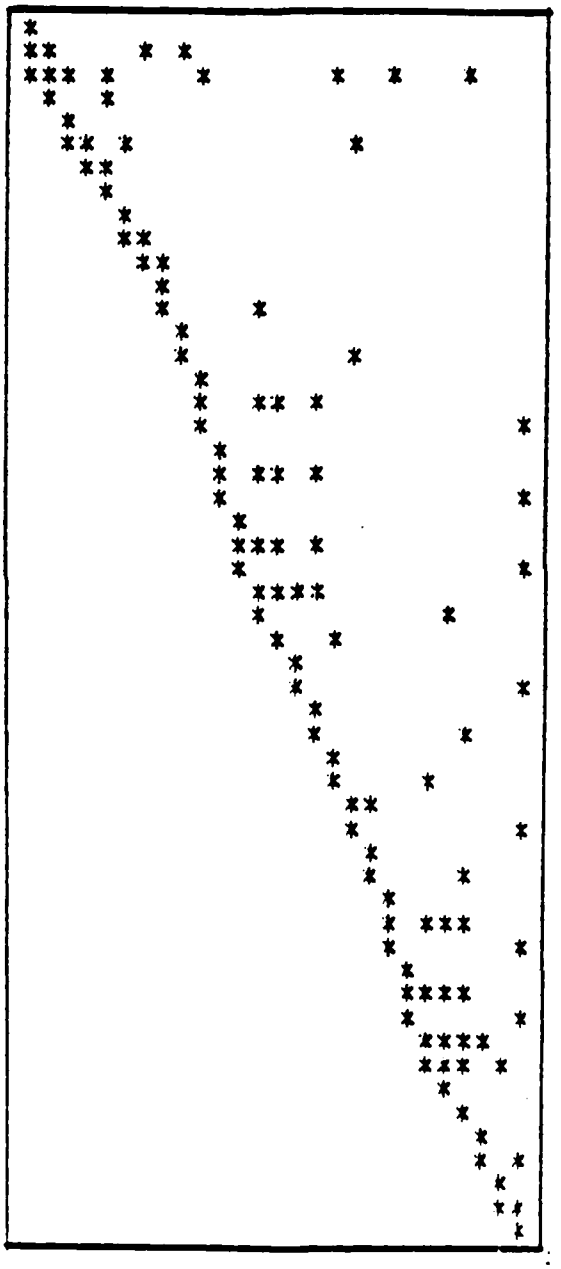


Figure 1. Permuted Nonzero Structure — AFIRO

ENCLOSURE D
"ON STEEPEST ASCENT WITH EQUALITY CONSTRAINTS"

J. M. Danskin

incomplete paper, 1986

ON STEEPEST ASCENT WITH EQUALITY CONSTRAINTS

by

John M. Danskin (Arlington Virginia)

ABSTRACT

The paper gives an algorithm for finding steepest ascent of a linear form in R^p, subject to q constraints, in at most q - 1 steps. It requires the storage of a q x q dense matrix. There is an application to linear programming.

§ 1. Introduction

Our problem is to maximize $z \cdot y$ subject to the constraints

$$|y| = 1 \tag{1.1}$$

and

$$b_j \cdot y = 0 , \quad j \in J . \tag{1.2}$$

Here z , y , and the b_j are in R^p , $z \neq 0$, and J is a finite nonempty set of indices. We denote by Γ the set of $y \in R^p$ satisfying (1.1)-(1.2). We will also call such a y "admissible".

Our method in the general case is to set up a linear manifold $\mathcal{L}$ in R^p and then to find the point $c \in \mathcal{L}$ closest to the origin.

We deal in § 2 with a trivial exceptional case. In § 3 we construct the linear manifold $\mathcal{L}$ and explain the significance of c. In § 4 we show how to find c in finitely many steps. In

§ 5 we give the easy estimate for computation time. In §§6-8 we note an application to linear programming. In § 10 we recall Lemke's closest-point principle [3] and the connection of our method with it.

§ 2. The exceptional case

We denote by J^0 the set of $j \in J$ for which $z \cdot b_j = 0$ and write $J^1 = J - J^0$. The exceptional case is the case $J^0 = J$: all the b_j are orthogonal to z. In that case the vector

$$v = z / |z| \tag{2.1}$$

satisfies the conditions (1.1) and (1.2), and $z \cdot v = |z|$. For any unit $v' \in R^p$ whatever, we have $z \cdot v = |z| \, v \cdot v' \leq |z|$, with equality holding only if $v' = v$. Hence v uniquely solves the problem (1.1)-(1.2), and the maximum is $\lambda = |z|$.

§ 3. The general case

Now J^1 is not empty. Suppose first that also J^0 is not empty. The vectors b_j for $j \in J$ now generate a cone B^0 in R^p. Using the Gram-Schmidt orthogonalization process, we construct an orthogonal basis $\{e_k\}_{k \in K}$ for B^0. Put

$$b_j' = b_j - \sum_{k \in K} (b_j \cdot e_k) e_k / e_k^2 \quad . \quad j \in J^1 . \tag{3.1}$$

Put

$$x^j = \frac{b_j'}{(z \cdot b_j')} \qquad j \in J^1 . \tag{3.2}$$

We will suppose from now on that J^1 has q elements. Denote by $\mathfrak{Z}$ the hyperplane in R^q given by

$$\sum_{j \in J^1} z_j = 1 . \tag{3.3}$$

The linear manifold $\mathcal{L}$ is the image of $\mathfrak{Z}$ in R^p given by the mapping

$$x = \sum_{j \in J^1} z_j x^j \quad . \quad z \in \mathfrak{Z} \quad . \tag{3.4}$$

We note three cardinal properties of $\mathcal{L}$. First

$$z \cdot x = 1 \quad \underline{\text{for all}} \quad x \in \mathcal{L}. \tag{3.5}$$

This follows from (3.2) and (3.4). Second,

$$x \cdot v' = 0 \quad \underline{\text{for all}} \quad x \in \mathcal{L} \quad \underline{\text{and}} \quad v' \in \Gamma. \tag{3.6}$$

It suffices to prove that $b_j' \cdot v' = 0$ for the b_j' of (3.1) and any admissible v'. But this follows because any admissible v' is by definition normal to the b_j for $j \in J^1$, which takes care of the first term in the right side of (3.1). As to the second, it is by definition normal to the b_j for $j \in J^0$ hence normal to B_0, and hence normal to each element e_k of the orthogonal basis. The third property is that

$$x \cdot b_j = 0 \quad \underline{\text{for all}} \quad x \in \mathcal{L} \quad \underline{\text{and}} \quad j \in J^0. \tag{3.7}$$

This is so because all the b_j' of (3.1) are orthogonal to B^0.

Now let c be the point on $\mathcal{L}$ closest to the origin. Then $\mathcal{L}$ has relative to c the following fourth, obvious property:

$$c \cdot (x - c) = 0 \quad \underline{\text{for all}} \quad x \in \mathcal{L}. \tag{3.8}$$

Now put

$$h = z - c/c^2 \ . \tag{3.9}$$

It may be that $h = 0$. Then since $c \in \mathcal{L}$ satisfies (3.6), $z \cdot \gamma' = 0$ for all admissible γ' and there is no direction of constrained ascent at all.

Otherwise $h \neq 0$. Then put

$$\lambda = |h| \ , \quad \gamma = h/\lambda \ . \tag{3.10}$$

Then (1.1) is satisfied. As to the constraints in (1.2) with $j \in J^0$, $z \cdot b_j = 0$ by the definition of J^0, and $b_j \cdot c = 0$ by (3.7). As to the constraints in (1.2) with $j \in J^1$, we have

$$x^j \cdot h = x^j \cdot z - x^j \cdot c/c^2 = 1 - 1 = 0 \tag{3.11}$$

by (3.5) and (3.8), so that, by the definition (3.2),

$$b_j' \cdot h = 0 \ , \quad j \in J^1 \ . \tag{3.12}$$

But since the e_k in (3.1) are orthogonal to both z and c, (3.12)

implies that

$$b_j \cdot h = 0 , \quad j \in J^1 . \tag{3.13}$$

Hence all the conditions (1.2) are satisfied, and $\gamma \in \Gamma$.

Now suppose γ' is any admissible vector. Then $c \cdot \gamma' = 0$ by (3.6), so that

$$z \cdot \gamma' = (z - c/c^2) \cdot \gamma' = \lambda \gamma \cdot \gamma' . \tag{3.14}$$

Hence γ yields the unique maximum over Γ, and that maximum is λ.

The case in which J^0 is empty is somewhat simpler. The subtracted projections in (3.1) are now gone. $b'_j = b_j$ for all $j \in J^1 = J$, and the rest of the construction goes through as before.

We now show how to find c, by an iteration with at most Min $\{p, q-1\}$ moves.

§ 4. Location of the closest point c

In what follows we suppose that the number q of elements in J^l is at least 3, the cases q=1,2 being obviously trivial. And we assume that $p \geq 2$.

The first basic element in our algorithm is steepest descent in $\mathcal{B}$. We recall what this is. For $z \in \mathcal{B}$ put

$$Q(z) = \sum_{i=1}^{p} \Big[\sum_{j \in J^l} z_j x_i^j \Big]^2 . \tag{4.1}$$

This is the distance-squared from the point $x \in \mathcal{L}$ given by (3.4) to the origin in R^p. Q has the partials

$$\frac{\partial Q(z)}{\partial z} = 2x \cdot x^j , \quad j \in J^l . \tag{4.2}$$

Form the direction numbers

$$gz_j = x \cdot (\bar{x} - x^j) , \quad j \in J^l , \tag{4.3}$$

in which $\bar{x}$ is the "barycenter"

$$\bar{x} = \frac{1}{q} \sum_{j \in J^1} x^j \quad . \tag{4.4}$$

If the gz_j are all zero, x is already the closest point c. Otherwise the direction number vector gz indicates the direction of steepest descent in $\mathcal{F}$, and the distance-squared is in fact strictly decreasing in that direction.

The second basic element in our algorithm is based on a simple geometric observation, as follows. Suppose that ℓ is any line lying in $\mathcal{L}$. Then the point x closest to the origin on ℓ is also the point closest to c on that line. Hence both the origin and the closest point c lie in the plane Π normal to ℓ at x. With this in mind, we will in our algorithm restrict the subsequent search to Π.

The algorithm can be started anywhere. We start it at the point $z^0 \in \mathcal{F}$ with $z_j^0 = 1/q$, $j \in J^1$. The image of z^0 under (3.4) is $\bar{x}$. If $gz = 0$, $\bar{x} = c$ and we are through. Otherwise denote by gx the image of gz under (3.4). Since gz corresponds to a direction of strict descent, then also $gx \neq 0$. We normalize gx to a unit direction vector gx^0. Now pass a line ℓ^1 through $\bar{x}$ in the direction gx^0. Let $x^{\#1}$ be the point

---

*) The $\#$ sign distinguishes this point from the "vertex" x^1 defined at (3.2).

---

on ℓ^1 closest to the origin. Denote by π^1 the plane normal to ℓ^1 (and therefore to gx^0) at $x\#^1$. From the construction, $x\#^1$ has a natural inverse in $\mathcal{G}$ under (3.4). We denote that inverse by z^1.

Now suppose that $1 \leq n < \text{Min}\{p, q-1\}$, and that we have arrived at a point $z^n \in \mathcal{G}$ and corresponding point $x\#^n \in \mathcal{L}$ under (3.4). Suppose that we have constructed directions $gx^0, \ldots, gx^{n-1}$, each normal to all those preceding it, and planes $\pi^0, \ldots, \pi^n$, normal respectively to the $gx^0, \ldots, gx^{n-1}$, and all containing c. Finally, we suppose that $x\#^n$ lies in all those planes.

Now construct the direction numbers gz_j, $j \in J^1$, from (4.3) with x replaced by $x\#^n$. If these are all zero, $x\#^n = c$ and we are finished. Otherwise, gz corresponds to a direction of strict decrease. Therefore its image gx under (3.4) is also nonzero, and corresponds to a direction of strict decrease on $\mathcal{L}$. Now we need to prove a technical point.

Denote by S the space spanned by the vectors $gx^0, \ldots, gx^{n-1}$. Introduce into S a coordinate system, with origin O at $x\#^n$ and with axes $OY^1, \ldots, OY^n$ parallel to the $gx^0, \ldots, gx^{n-1}$. Now let u be any unit vector in S with components $u_1, \ldots, u_n$ along those axes. Consider the ray t issuing from $x\#^n$ in the

direction of u, with s measuring the distance from $x\#^n$. Now the vectors $gx^0, \ldots, gx^{n-1}$ are orthogonal not only to one another but also to the vectors $x\#^n - c$ and c, which are themselves orthogonal. Hence the distance-squared to the origin from the point on ℓ corresponding to s is

$$D^2_u(s) = c^2 + (x\#^n - c)^2 + s^2[u_1^2 + \ldots + u_n^2]$$

$$= (x\#^n)^2 + s^2 . \qquad (4.5)$$

Hence the directional derivative of the distance-squared in the direction u, at $x\#^n$, is zero. This is the technical point we needed.

Since the vector gx defined above is nonzero and corresponds to a direction of strict decrease for the distance-squared, it does not lie in S. We now put

$$(gx)' = gx - (gx \cdot gx^0)\,gx^0 - \ldots - (gx \cdot gx^{n-1})\,gx^{n-1} . \qquad (4.6)$$

Since gx does not lie in S, $(gx)' \neq 0$.

We now normalize $(gx)'$ to a unit direction vector gx^n and pass a line ℓ^{n+1} through $x\#^n$ in the direction gx^n.

Denote by $x\#^{n+1}$ the point on ℓ^{n+1} closest to the origin. Denote by π^{n+1} the plane normal to ℓ^{n+1} (and therefore to gx^n) at $x\#^{n+1}$. Let $z^{n+1} \in \mathcal{G}$ be the "natural inverse" of $x\#^{n+1}$.

We see that all the induction hypotheses made at the beginning of this general step with $n \geq 1$ are fulfilled. We assumed that all the planes $\pi^1, \ldots, \pi^n$ contained c. Now π^{n+1} does as well, by its construction. And, since $x\#^n$ lay in all the $\pi^1, \ldots, \pi^n$ and the direction gx^n of ℓ^{n+1} is orthogonal to all the normals $gx^0, \ldots, gx^{n-1}$ to those planes, then also $x\#^{n+1}$ lies in all the $\pi^1, \ldots, \pi^n$. And it lies in π^{n+1} by the definition of π^{n+1}. So we are ready to proceed to the next step.

The dimension of $\mathcal{L}$ cannot exceed q-1 because there were only q-1 vectors, $x^2 - x^1, \ldots, x^{q-1} - x^1$, in the original basis. And it cannot exceed p, because $\mathcal{L}$ is imbedded in R^p. Hence put

$$r = \text{Min } \{ p, q-1 \} . \tag{4.7}$$

The dimension of $\mathcal{L}$ then does not exceed r. Since we add one element to the orthonormal basis $\{gx^0, \ldots\}$ on each move, the number of moves cannot exceed r, and the closest point c will be reached exactly on the last move. □


DATE
FILMED
28